Titre: Title:	Génération automatique d'horaires en milieu hospitalier
Auteur: Author:	Stéphane Bourdais
Date:	2003
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Bourdais, S. (2003). Génération automatique d'horaires en milieu hospitalier [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7228/

UNIVERSITÉ DE MONTRÉAL

GÉNÉRATION AUTOMATIQUE D'HORAIRES EN MILIEU HOSPITALIER

STÉPHANE BOURDAIS
DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRE ÈS SCIENCES APPLIQUÉES (M.Sc.A.)
(GÉNIE ÉLECTRIQUE)
Septembre 2003

Your file Votre référence
ISBN: 0-612-89181-X
Our file Notre référence
ISBN: 0-612-89181-X

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

GÉNÉRATION AUTOMATIQUE D'HORAIRES EN MILIEU HOSPITALIER

présenté par: Stéphane BOURDAIS

en vue de l'obtention du diplôme de: Maître ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de:

GAGNON Michel, Ph.D, président
PESANT Gilles, Ph.D, membre et directeur de recherche
GALINIER Phillipe, Ph.D, membre et codirecteur de recherche
JAUMARD Brigitte, T. Doct., T. Hab., membre et codirectrice de recherche
SORIANO Patrick, Ph.D, membre

REMERCIEMENTS

Je tiens tout d'abord à remercier mes directeurs Gilles Pesant, Philippe Galinier et Brigitte Jaumard, pour leur présence, leurs conseils, leur soutien autant moral que financier et le gros travail de correction qu'ils ont effectué sur ce mémoire. Je tiens à remercier particulièrement Gilles pour son travail sur les contraintes globales et son expérience de la programmation par contraintes et Philippe pour son expérience dans le domaine des heuristiques.

Je remercie l'équipe IRIS, constituée de Pascal Labit, Jean-Philippe Doyon et Jacques-André Delafosse, pour le partage de connaissances sur le problème des unités de soins infirmiers.

Je remercie l'équipe MEDHOR, consituée de Salim Hamid et Nourredine Hail, pour le travail acharné qu'ils ont fourni pour me concevoir des fichiers de données réalistes pour les hôpitaux Sacré-Coeur et Santa-Cabrini.

Je remercie tous les gens que j'ai pu côtoyer au CRT, au GERAD et à l'École Polytechnique pendant ma maîtrise et plus particulièrement mes collègues de bureau, Schnouff, Cathy, Redoine et Imotep, pour m'avoir supporté au cours de ces deux années.

Un grand merci pour ma famille qui, même de l'autre côté de l'atlantique, n'a cessé de m'encourager.

Enfin j'ai une pensée toute particulière pour mes colocs et amis, Anne-va, Lolo, Rémy et Hugo, le coloc à temps partiel.

RÉSUMÉ

Ce mémoire de maîtrise présente un modèle de programmation par contraintes ainsi que des stratégies de recherche permettant de formuler et de résoudre des problèmes de confection d'horaires en milieu hospitalier. Ce problème a déjà été largement étudié et beaucoup d'approches différentes ont été développées au cours des années, mais elles sont souvent trop liées à leur contexte de développement. Concevoir une méthode robuste pouvant s'appliquer à différents contextes reste donc un défi à relever.

Nous décrivons une méthode flexible et plutôt générale de programmation par contraintes qui intègre des heuristiques de recherche de façon originale. Nous comparons quelques stratégies sur un ensemble de données hétérogènes. Dans le cadre de ce mémoire, nous nous sommes concentrés sur la satisfaction plutôt que l'optimisation, mais nous indiquons les directions à suivre pour transformer notre méthode en une méthode d'optimisation. Des horaires réalisables sont trouvés en moins de cent secondes.

Mots Clefs : confection d'horaire, milieu hospitalier, programmation par contraintes, heuristiques de recherche

ABSTRACT

This master's thesis presents a constraint programming model and search strategies to formulate and solve staff scheduling problems in health care. This is a well-studied problem for which many different approaches have been developed over the years but it remains a challenge to apply the methods developed to the various contexts encountered.

We describe a flexible and rather general constraint programming approach that combines search heuristics in an original manner. We also compare several strategies to solve the problem on a heterogeneous sample of data. In this master's thesis, we concentrate on satisfaction rather than optimization, but we give information on how to transform our method in an optimization method. We obtain feasible schedules in less than one hundred seconds.

Key Words : staff scheduling, health care, constraint programming, search heuristics

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ALGORITHMES

LISTE DES SIGLES ET ABRÉVIATIONS

CSOP	Constraint Satisfaction Optimization Problem
CSP	Constraint Satisfaction Problem
HCOP	Hierarchical Constraint Optimization Problem
HIBISCUS	Horaire d'Infirmières Basé Sur les Contraintes
PC	Programmation par contraintes
PCSP	Partial Satisfaction Problem

LISTE DES ANNEXES

INTRODUCTION

Au cours des trente dernières années, le problème de confection d'horaires de travail a bénéficié de beaucoup d'attention de la part de la communauté scientifique aussi bien que de l'industrie. Pour s'en convaincre il suffit de voir le nombre d'articles publiés sur le sujet ou le nombre de logiciels commerciaux existants (voir [14, 24, 2, 48, 30, 23, 3, 12, 42, 35] pour quelques références). C'est une tâche à laquelle sont confrontées la plupart des organisations. Les horaires générés doivent respecter le droit du travail et les conventions collectives. Ils ont un impact direct sur l'organisation du travail et notamment sur les coûts de personnel mais aussi de gestion du personnel. Nous allons nous intéresser plus particulièrement aux problèmes de confection d'horaires pour médecins et infirmiers. Les difficultés rencontrées sont multiples. La première est l'aspect fondamentalement combinatoire du problème, ce qui ne permet pas, dans la plupart des cas, de trouver rapidement des solutions acceptables, même par un expert. À cela vient s'ajouter le facteur humain :

- l'absentéisme implique souvent de modifier fréquemment les horaires prévus initialement,
- plusieurs études sur les conditions de travail du personnel (dont celles de Knauth [26, 27]) ont montré la nécessité de respecter certaines règles complexes, dites ergonomiques, et enfin,
- il est important de maintenir un certain niveau d'équité parmi le personnel pour éviter tout sentiment d'injustice.

Comme nous l'avons mentionné plus haut, il existe déjà beaucoup de solutions, dont certaines sont utilisées en pratique. Cependant, soit il s'agit d'un logiciel développé spécifiquement pour un contexte particulier, soit il s'agit d'un logiciel distribué à grande échelle ne permettant pas de tenir compte de toutes les spécificités que l'on peut rencontrer. Il reste donc de nombreux hôpitaux pour lesquels aucun logiciel

actuel ne convient parfaitement. Nous proposons une approche qui prend en compte un ensemble de règles très riche, permettant d'apporter des solutions à ces hôpitaux. Nous utilisons pour cela la programmation par contraintes (PC) qui est un outil issu conjointement des communautés de l'intelligence artificielle et de la recherche opérationnelle. Il existe déjà plusieurs travaux sur la confection automatisée d'horaire utilisant la PC. Nous apportons tout de même quelques nouveautés :

- nous utilisons systématiquement des contraintes globales lorsque cela est possible,
- nous proposons des heuristiques de recherche modulaires capable de s'auto-adapter au modèle et, enfin,
- la méthode se veut robuste dans la résolution de plusieurs contextes differents.

Le projet a débuté avec les travaux de William Feuilletin [19] en 2000, suivis de ceux de Julien Félisiak en 2001. Ces travaux ont aboutit à une première version du prototype **HIBISCUS** (Horaires d'Infirmières Basés Sur les Contraintes) traitant uniquement un contexte particulier aux unités de soins infirmiers. Utilisant seulement des stratégies de recherche standards, cette première version ne donnait cependant pas de résultat satisfaisant.

Le modèle de programmation par contrainte sur lequel est basée la présente version de **HIBISCUS** s'inspire du précédant, mais il est plus général et utilise mieux les contraintes globales. De plus, il intègre des stratégies de recherche plus élaborées qui permettent de trouver des solutions en quelques secondes, et ce pour des contextes hétérogènes.

Organisation du mémoire

Dans le premier chapitre nous présentons de façon générale le problème de confection d'horaires. Nous donnons une nomenclature des règles ainsi que quelques critères permettant d'évaluer la qualité d'une méthode. Enfin, nous présentons une revue de la littérature sur le sujet en nous concentrant sur les méthodes de la programmation mathématique, de la programmation par contraintes, des méta-heuristiques et de quelques méthodes hybrides, qui constituent la grande majorité.

Dans le second chapitre, nous présentons de manière informelle le problème que nous modélisons par la suite. Nous décrivons les caractéristiques du personnel, des

tâches et d'un ensemble de règles générales permettant de modéliser des environnements complexes.

Le chapitre trois survole les concepts fondamentaux de la PC en soulignant les points importants pour une bonne modélisation. Nous présentons les algorithmes participant à la résolution des problèmes et discutons de leur complexité. Enfin, nous y introduisons quelques notations utiles pour les deux chapitres suivants.

Le chapitre quatre présente le modèle de PC du problème. Après une description du langage de modélisation utilisé, nous définissons les variables puis donnons l'expression des contraintes correspondant à chaque règle du chapitre deux. À chaque fois, nous discutons de la complexité des algorithmes de filtrage engendrés.

Les stratégies de recherche sont décrites au chapitre cinq. Plus qu'une stratégie, nous décrivons un processus permettant de créer plusieurs stratégies en utilisant la structure particulière des problèmes de confection d'horaire. Il s'agit en fait d'un modèle de stratégie que l'on peut instancier de plusieurs façons dépendamment des spécificités du problème concret étudié. D'autre part ces stratégies ont la particularité de s'auto-adapter par l'intermédiaire d'heuristiques *d'incitation* que l'on peut associer à certaines contraintes du modèle.

Le chapitre six présente les données utilisées et les résultats obtenus. Les contextes de plusieurs hôpitaux de la région de Montréal y sont décrits. Cela inclut la description du personnel, des tâches ainsi que la liste des règles prises en compte. Enfin, nous comparons plusieurs stratégies et discutons des résultats obtenus.

CHAPITRE 1

PRÉSENTATION DU PROBLÈME ET REVUE DE LA LITTÉRATURE

1.1 Le problème de confection d'horaires

La gestion du personnel hospitalier est une recherche perpétuelle d'un équilibre entre trois objectifs contradictoires : la qualité des soins prodigués, les conditions de travail du personnel et les coûts financiers. D'après Warner [47], trois niveaux de décision interviennent dans le processus de gestion du personnel infirmier :

i. **Le niveau stratégique** (*"The staffing decision"*). C'est l'étape du dimensionnement des unités de soins en réponse à l'évaluation de la demande. Au cours de cette étape, on détermine pour chaque catégorie du personnel les effectifs nécessaires pour pouvoir couvrir la demande.

ii. **Le niveau tactique** (*"The scheduling decision"*). C'est l'étape de la confection d'horaires à proprement parler, au cours de laquelle on détermine la nature, le nombre et les dates des tâches affectées à chaque membre du personnel. L'horizon de planification peut s'étaler sur une période allant d'une semaine à plusieurs mois.

iii. **Le niveau opérationnel** (*"The allocation decision"*). Ce sont des décisions prises au jour le jour pour réagir aux événements imprévus tels que les absences ou les fortes hausses de la demande.

Dans cette étude, nous nous intéressons seulement au second niveau de décision, ce qui simplifie beaucoup le problème car le nombre d'unités de soins, leurs effectifs ainsi que la demande ont été estimés au premier niveau de décision. Les coûts financiers étant principalement déterminés lors du dimensionnement des unités de soins, il ne reste vraiment que deux objectifs pour l'étape de confection des horaires : la qualité des soins et les conditions de travail du personnel.

Plus concrètement, la confection d'horaires est un problème d'affectation d'un ensemble de ressources (le personnel) à un ensemble de tâches (appelés aussi quarts) sujet à certaines contraintes (règles). Concevoir un horaire, c'est donc trouver une affectation des ressources aux différentes tâches qui satisfait les contraintes. Souvent, il en existe beaucoup, et on s'intéresse alors à trouver un ensemble de bons horaires relativement à certains critères.

L'une des particularités du milieu hospitalier est que l'on rencontre souvent des problèmes sur-contraints (à cause de l'hétérogénéité des règles, surtout celles concernant les préférences individuelles). Il faut alors résoudre un problème supplémentaire : le choix des contraintes que l'on va se permettre de violer afin d'obtenir une solution acceptable. La plupart des approches définissent un ensemble de contraintes *dures*, qu'il est impératif de respecter, et un ensemble de contraintes *souples*, que l'on se permet de violer.

Dans la littérature, chaque problème étudié se base sur un contexte particulier. Il en résulte une multitude de modèles, chacun ayant ses propres ensembles de contraintes dures et souples ainsi que son propre objectif. Il y a plusieurs raisons à cela : la législation propre à chaque état, les habitudes propres à chaque hôpital, voire chaque département ou unité de soins, ainsi que des conditions socio-économiques particulières comme la forte pénurie de personnel que subissent actuellement les hôpitaux canadiens. Malgré cela, on trouve beaucoup de similarités dans tous ces modèles. Quelques travaux récents[10], dont celui présenté dans ce mémoire, cherchent à spécifier des modèles généraux pouvant s'appliquer à plusieurs contextes en s'appuyant sur une nomenclature précise des règles et des objectifs. Après examen des travaux de Carter et Lapierre [10], de Forget [20] et de nombreux autres (voir section 1.3), voici la nomenclature qui a été retenue :

[DEM] *respect de la demande,*

[WOR] *respect des charges de travail individuelles,*

[AVA] *disponibilités,*

[ERG] *règles ergonomiques,*

[FAI] *règles d'équité.*

La liste des objectifs rencontrés est assez longue. Voici les principaux (dans la majorité des cas, plusieurs sont pris en compte simultanément) :

- *minimiser l'effectif du personnel,*
- *minimiser les coûts économiques,*
- *minimiser les périodes de sur-effectif et de sous-effectif,*
- *minimiser les écarts entre la charge de travail voulue et la charge de travail réelle,*
- *équilibrer les quarts d'un certain type de façon équitable,*
- *maximiser la satisfaction des préférences individuelles.*

1.2 Les méthodes de confection d'horaires

Warner [47] donne une liste de critères permettant de comparer des méthodes de confection d'horaires :

Couverture : mesure de la qualité de la couverture des tâches. Cet indice est directement relié à la qualité des soins prodigués. Si les horaires générés ne couvrent pas toutes les tâches, l'hôpital est obligé de faire appel à du personnel supplémentaire afin d'assurer un service minimum.

Qualité : mesure de la qualité des horaires individuels générés, *i.e.* , des conditions de travail du personnel. Elle doit tenir compte de la législation en vigueur, de certaines règles propres à l'hôpital ou au service, et des préférences et aversions du personnel.

Stabilité : mesure du degré avec lequel le personnel peut connaître ses affectations futures, en particulier les jours de congé et les vacances, et du degré de confiance dans la méthode. Plus la méthode est stable, plus le personnel est capable de prévoir son horaire avec suffisamment de précision pour pouvoir organiser son temps.

Flexibilité : mesure de la capacité de la méthode à faire face aux changements d'une période à l'autre (choix des vacances, préférences individuelles, passage de temps complet à temps partiel, changement de personnel).

Équité : mesure du degré de confiance dans l'équité de la répartition des tâches par le système.

Coût : mesure de l'ensemble des ressources mises en oeuvre pour la conception de l'horaire.

Les méthodes de confections d'horaires utilisées dans les hôpitaux diffèrent suivant le type d'horaire généré. Il existe trois grands types d'horaires : les horaires cycliques avec rotations, les horaires cycliques sans rotation et les horaires non cycliques.

1.2.1 Horaire cyclique avec rotations

Un horaire cyclique avec rotations est construit à partir de patrons d'horaires sur une période courte, que les membres du personnel suivent à tour de rôle. Ainsi, après une rotation complète du personnel, chacun des membres a fait exactement les mêmes quarts et dans le même ordre, ce qui garantit une équité parfaite. Cette méthode est en outre très stable, car l'horaire peut se répéter sur une longue durée. Cependant, elle n'est pas du tout flexible car elle considère que le personnel est homogène et ne permet pas de prendre en compte des préférences individuelles (disponibilités, vacances, règles ergonomiques, etc.).

1.2.2 Horaire cyclique sans rotation

Un horaire cyclique sans rotation est également construit à partir de patrons d'horaires sur une courte période mais, cette fois, chaque membre du personnel répète son propre horaire. L'équité peut être conservée à condition que les patrons d'horaires soient bien construits, *i.e.* , avec une bonne distribution de tous les types de quart. Le grand avantage par rapport à la méthode précédente est la possibilité de personnaliser les horaires, ce qui donne déjà un peu plus de flexibilité à la méthode. Mais le fait que le même horaire soit répété en boucle ne permet pas de prendre en compte des préférences épisodiques comme des périodes de vacances ou des indisponibilités occasionnelles.

1.2.3 Horaire non cyclique

Les horaires non cycliques sont reconstruits au début de chaque période de planification. C'est la méthode la plus flexible car elle permet de prendre en compte

beaucoup de préférences individuelles. L'équité est un peu plus difficile à respecter, mais peut néanmoins l'être si certaines règles de distribution sont respectées. De plus, si le personnel est hétérogène (des personnes plus anciennes, plus expérimentées, etc.), l'équité est souvent difficile à définir.

1.3 Revue de la littérature

Nous nous intéressons uniquement aux méthodes fournissant des horaires non cycliques permettant de prendre en compte les requêtes individuelles du personnel. Dans ce contexte particulier, essayer de déterminer l'effectif minimum du personnel pouvant couvrir la demande est trop complexe. Les travaux de Burns et Carter [6] ne s'appliquent qu'aux horaires cycliques et au prix d'hypothèses bien trop restrictives pour le projet actuel. Les premiers travaux datent des années 70 avec Warner [47] et Miller *et al.* [33]. Parmi les approches récentes, on trouve des méthodes utilisant la programmation mathématique (section 1.3.2), des méta-heuristiques (section 1.3.3) et la programmation par contraintes (section 1.3.4) ainsi que quelques approches hybrides (section 1.3.5). D'autres méthodes, moins utilisées, sont mentionnées à la section 1.3.6.

1.3.1 Travaux précurseurs

Pour Warner [47], les solutions réalisables sont celles qui respectent la demande et les charges de travail. Un certain nombre de règles souples servent à mesurer la qualité des horaires individuels. Le personnel doit déterminer individuellement des poids pour chacune de ces règles, il s'agit principalement de préférences ou d'aversions pour les quarts et les congés isolés, les séquences de travail de plus de six jours et pour certains enchaînements de quarts. L'équité se traduit par une règlementation stricte du nombre de points que chaque membre du personnel peut affecter à l'ensemble des critères avant la planification.

La résolution est heuristique et comporte deux phases : une phase d'initialisation pour trouver une solution réalisable et une phase ressemblant à une recherche locale au cours de laquelle l'objectif est amélioré itérativement. La méthode obtient des résultats pour des unités de soins allant jusqu'à 47 infirmières sur des horizons de planification de 42 jours (découpés en périodes de 14 jours). Il faut noter tou-

tefois que la politique d'affectation des week-ends est pré-déterminée, ce qui réduit significativement la complexité du problème et permet aussi de découper l'horizon de planification en sous-périodes de 14 jours sans que le couplage entre les sous-périodes ne soit trop fort.

Au même moment, Miller *et al.* [33] s'est intéressé au problème de l'affectation des jours de repos (dans lequel on cherche à dissocier les périodes de repos et les périodes de travail sans s'intéresser à leur nature). L'approche proposée ici est à la base des méthodes plus récentes de programmation mathématique généralisée basées sur la technique de génération de colonnes. Les variables ne sont pas des quarts, mais des horaires individuels au complet. Pour chaque membre du personnel, on énumère l'ensemble des horaires réalisables (*i.e.* , respectant la charge de travail, les disponibilités et quelques contraintes ergonomiques dures). Un poids est associé à chacun des horaires en fonction de diverses règles.

L'algorithme de résolution est une heuristique en deux phases : une phase d'initialisation et une phase d'amélioration itérative d'une somme pondérée du poids global de l'horaire et de pénalités traduisant la non-satisfaction de la demande (périodes de sur- et sous-effectif). La méthode a obtenu des résultats sur des problèmes composés d'une douzaine d'infirmières sur des horizons de 28 jours. Le point faible de cette méthode est certainement l'énumération initiale des horaires individuels ainsi de leur stockage explicite qui prend beaucoup de place. De plus, même si le système des poids peut sembler très flexible, il est néanmoins fastidieux en pratique car les horaires individuels sont nombreux.

1.3.2 Programmation mathématique

Forget [20] et Beaulieu [4] proposent un modèle PLNE (Programmation Linéaire en Nombres Entiers) prenant en compte des contraintes ergonomiques complexes qu'ils appliquent aux cas de deux hôpitaux de Montréal : l'Hôpital Sacré-Coeur et l'Hôpital Santa-Cabrini. Les contraintes souples sont traduites par des pénalités dont la somme pondérée constitue un objectif à minimiser. Malgré cela, leurs problèmes sont sur-contraints dans la plupart des cas. Ils proposent donc une procédure automatique qui relaxe successivement certaines contraintes dures.

L'algorithme de résolution utilise CPLEX. Cette approche fournit des horaires pour des salles d'urgence composées d'environ 20 médecins sur des horizons de pla-

nification d'un mois. Les sections 6.1.5 et 6.1.6 décrivent en détail les contextes des deux hôpitaux mentionnés plus haut.

Jaumard *et al.* [25] proposent un modèle général et flexible de programmation mathématique généralisée utilisant la technique de génération de colonnes. La décomposition utilisée est semblable à celle de Miller *et al.* [33] mais, dans ce cas, les horaires individuels sont générés au cours de la recherche. Le problème maître est un PLNE qui cherche la configuration optimale des horaires individuels déjà générés relativement à un objectif tenant compte de critères tels que la satisfaction de la demande, la qualité et l'équité de l'horaire. À chaque itération un problème auxiliaire génère un ou plusieurs horaires individuels (colonnes dans la matrice des contraintes du problème maître) susceptibles d'améliorer la solution globale. L'algorithme de résolution du problème maître utilise CPLEX tandis que le problème auxiliaire utilise un algorithme de programmation dynamique combiné à un branch-and-bound.

Cette méthode fournit des résultats pour des unités de soins d'environ 50 infirmières sur des horizons de planification de six semaines.

Caprara, Monaci et Toth [9] proposent plusieurs méthodes de programmation mathématique dont une basée sur la génération de colonnes mais dans un contexte bien plus restrictif que le précédent. D'une part, le personnel est indifférencié et, d'autre part, la modélisation choisie interdit tout enchaînement de quarts différents (les blocs de travail étant nécessairement homogènes). De plus la charge de travail du personnel est exprimée en nombre de blocs de chaque type, ce qui est assez peu conventionnel.

La situation étant considérablement simplifié, la méthode fournit des horaires pour des exemplaires de problème impliquant plus de 1000 personnes sur des horizons allant jusqu'à 182 jours.

1.3.3 Méta-heuristiques

Une méta-heursitique est un schéma algorithmique de haut niveau pouvant se spécialiser pour résoudre des problèmes d'optimisation de façon approchée. Certaines s'inspirent de concepts scientifiques provenant de domaines aussi divers que la biologie (colonies de fourmies), la génétique (algorithmes génétiques) ou la thérmodynamique (recuit simulé).

Thompson [43] propose une méthode de recuit simulé pour résoudre un modèle basé sur celui de Dantzig [15]. La charge de travail et les disponibilités sont des contraintes dures. Les autres règles sont souples et sont exprimées par un ensemble de poids associés aux affectations potentielles. L'objectif est composé de pénalités de sur- et sous-effectif ainsi que de la somme des poids des affectations. Le principal défaut de cette approche réside dans la difficulté de déterminer les poids des affectations. Il est en outre impossible avec ce modèle d'exprimer des règles ergonomiques complexes.

Dowsland [17] s'intéresse au problème d'affectation des jours de repos et utilise, à l'instar de Miller [33], un modèle dans lequel les inconnues sont des horaires individuels réalisables. Les horaires individuels sont énumérés et évalués dans une phase d'initialisation. L'algorithme utilisé est une heuristique de recherche tabou particulière faisant intervenir deux phases en alternance : l'une visant à rendre la solution réalisable en terme de satisfaction de la demande, l'autre visant à minimiser le poids global de l'horaire.

Comme la méthode de Miller, la modélisation utilisée est assez flexible car elle permet de prendre en compte n'importe quelle règle ergonomique. Cependant l'énumération des horaires coûte cher et limite la taille des horizons de planification à une semaine ou deux au plus. De ce fait, il est plus difficile de prendre en compte des contraintes d'équilibrage à long terme.

Buzòn [7] propose également une heuristique de type recherche tabou utilisant diverses structures de voisinages. Le modèle permet de prendre en compte des contraintes ergonomiques complexes. Cette approche fournit des résultats pour des salles d'urgence d'environ 20 médecins sur des horizons de planification de trois mois. La section 6.1.7 décrit en détail le contexte étudié ici.

1.3.4 Programmation par contraintes

Beaucoup de travaux récents se sont tournés vers la programmation par contraintes (PC). Dans sa thèse, Heus [22] modélise le problème sous forme d'un CSP (Problème de Satisfaction de Contraintes). Il discute de la façon de briser certaines symétries dans le modèle et propose deux stratégies de recherche, l'une d'elle utilisant la décomposition *affectation des jours de repos / affectation des tranches horaires*. Ces travaux ont aidé à élaborer le logiciel commercial *Gymnaste* [14]. La méthode conçoit assez

facilement des horaires pour environ 30 personnes sur des horizons de deux semaines. Le défaut de cette approche est de ne pas modéliser des règles souples. Ce n'est pas gênant dans le contexte fourni en exemple car les problèmes sont tous cohérents.

Abdennadher et Schenker [1] proposent un modèle sous forme de PCSP (Problème de satisfaction de contraintes partiel) et distinguent un ensemble de contraintes dures et un ensemble de contraintes souples. La stratégie est similaire à celle de Heus [22] et comporte trois phases : l'affectation des jours de repos, l'affectation des quarts de nuits puis l'affectation des quarts de jour. La minimisation se fait sur le maximum des coûts des horaires individuels, ce qui permet d'éviter les solutions avec une trop grande variance. Les auteurs mettent l'accent sur l'interactivité de leur approche qui permet de voir rapidement l'impact des changements effectués par l'utilisateur sur le modèle. Les horaires sont générés pour 20 infirmières sur des horizons de planification d'un mois.

Les méthodes suivantes modélisent le problème sous forme de CSP ou de PCSP, comme dans les approches précédentes, mais utilisent en outre les contraintes souples sous forme d'heuristiques pour guider la recherche. Darmoni *et al.* [16] utilisent des compteurs mis à jour dynamiquement au cours de la recherche, Caseau *et al.* [11] se basent sur un calcul *d'entropie* pour chaque paire quart/jour et enfin Meisels *et al.* [31] utilisent des *règles d'affectation.* Les approches de Darmoni et Caseau fournissent des horaires pour des effectifs d'une dizaine de personnes et pour des horizons d'une semaine, celle de Meisels pour des effectifs de 20 personnes et des horizons de deux semaines.

Cheng et Lee [13] utilisent une modélisation double pour améliorer le filtrage. Dans leur approche, les contraintes souples sont traitées comme des disjonctions dans l'arbre de recherche, *i.e.* , un point de choix binaire par contrainte souple est ajouté à la racine de l'arbre, la contrainte devant être respectée uniquement dans la première branche. Le premier défaut de cette approche est le biais introduit par l'ordre de ces points de choix dans l'arbre, car les contraintes correspondant à des points de choix élevés seront rarement remises en cause. Pour régler le problème de l'équité, les auteurs proposent un ordonnancement faisant intervenir des paramètres aléatoires pour les contraintes souples exprimant les souhaits individuels. L'explosion combinatoire

de l'arbre de recherche lorsqu'on ajoute des contraintes souples est un autre défaut majeur de l'approche. L'approche est validée sur un exemple comprenant des règles assez complexes et surtout un grand nombre de types de quarts différents. Elle fournit des horaires pour des effectifs allant jusqu'à 27 infirmières sur des horizons d'une semaine avec 11 quarts différents.

Saadie [41] propose un modèle CSP pour résoudre les problèmes des salles d'urgence de l'Hôpital Sacré-Coeur de Montréal (voir aussi Forget [20] et Beaulieu [4]). L'heuristique de recherche utilisée est semblable à celle de Heus [22] et Abdennadher [1].

Enfin, Feuilletin [19] et Felisiak ont modélisé le même problème que Jaumard *et al.* [25] sous forme de CSP. Leur travail a servi de point de départ à cette maîtrise.

1.3.5 Méthodes hybrides

Plus récemment, des approches hybrides entre la PC et diverses heuristiques de recherche locale [8] ont été développées, la PC étant utilisée pour parcourir implicitement des voisinages de grande taille.

Rousseau, Pesant et Gendreau [38] présentent un algorithme génétique dont les populations à chaque itération sont constituées de solutions partielles qui sont ensuite complétées par le branch-and-bound de la PC. Pour modéliser les contraintes complexes, les auteurs utilisent deux contraintes génériques : la contrainte `DISTRIBUTE` et une nouvelle contrainte appelée *"PATTERN constraint"* modélisée avec Ilog OPL Studio. Les expérimentations ont été réalisées sur les mêmes problèmes que ceux de Forget [20] et montrent la supériorité de la méthode hybride sur une méthode de PC pure et une méthode de recherche locale pure.

Meyer [32] propose un modèle sous forme de HCOP (*Hierarchical Constraint Optimisation Problem*). En plus d'associer un poids à chacune des contraintes, un modèle HCOP définit une hiérarchie parmi les contraintes. Le but est de trouver une affectation qui satisfait toutes les contraintes dures (niveau 0) et qui minimise le poids des violations des contraintes de niveau inférieur, en tenant compte de la hiérarchie. Voici la hiérarchie des contraintes proposées par les auteurs :

0 : vacances et temps de repos minimal entre deux quarts (11h dans le contexte étudié),

1 : dernières contraintes ajoutées par l'utilisateur (la méthode est interactive),

2 : respect de la demande minimale,

3 : contraintes sur le nombre de jours de repos,

4 : respect de la demande,

5 : charges de travail individuelles spécifiées dans les contrats,

6 : préférences parmi un ensemble de patrons d'horaires de 15 jours,

7 : préférences et aversions individuelles.

La méthode proposée est une recherche locale utilisant la PC pour parcourir des voisinages de grande taille. La plupart des contraintes étant souples, l'auteur propose en outre une méthode de propagation spéciale basée sur les ensembles flous.

1.3.6 Autres méthodes

D'autres méthodes moins répandues sont également mentionnées dans la littérature. On peut trouver quelques références dans les travaux de Darmoni [16] et Heus [22]. Il s'agit de méthodes basées sur des systèmes experts ou des réseaux neuronaux.

CHAPITRE 2

DÉFINITION DU PROBLÈME ÉTUDIÉ

Dans ce chapitre, nous décrivons un problème général de confection d'horaires non cycliques en nous aidant des travaux de Carter et Lapierre [10] sur les salles d'urgences de six hôpitaux de la région de Montréal, de Jaumard et al. [25] et de Feuilletin [19] sur les unités de soins infirmiers de l'Hôpital Royal Victoria et de l'Hôpital général de Montréal, de Beaulieu [4] et Saadie [41] sur les salles d'urgences de l'Hôpital Sacré-Coeur, de Forget [20] sur les salles d'urgence de l'Hôpital Santa-Cabrini et de Buzòn [7] sur celles de l'Hôpital général juif. Le problème décrit est assez général pour pouvoir représenter chacun des contextes particuliers cités plus haut.

2.1 Le personnel

Il s'agit ici uniquement des médecins et des infirmiers ou infirmières. Les problèmes étudiés ne concernent qu'un seul type de personnel à la fois bien qu'il soit tout à fait envisageable d'utiliser la présente approche pour traiter des problèmes mixtes. On aura donc des problèmes pour le personnel infirmier et des problèmes pour les médecins. Voici la liste des attributs du personnel que **HIBISCUS** prend en compte (ils suffisent pour exprimer l'ensemble des règles rencontrées jusqu'à présent) :

- ◇ *l'ancienneté,*
- ◇ *le niveau de compétence,*
- ◇ *le statut temps complet/partiel* (exprimé par la charge de travail).

2.2 Les tâches

Une tâche est une période de travail indivisible au cours d'une journée pouvant être accomplie par une seule personne. Elle est caractérisée par un symbole, une heure de début, une heure de fin et éventuellement un nombre d'heure payée (s'il s'agit d'une activité payée).

L'ensemble des tâches comprend, en plus des quarts de travail, plusieurs activités reliées au travail qui nécessitent également d'être planifiées. Il s'agit concrètement de cours et de réunions. Ce sont des tâches plus rares que les quarts de travail et qui interviennent à peu près une fois par semaine.

Remarque 1 *: Dans la suite du mémoire, on utilisera le terme* quart *pour désigner une tâche. On parlera également de quart* `off` *pour un jour de congé et de quart* `vac` *pour un jour de vacances.*

Les types de tâches. Un type de tâche est un sous ensemble de tâches ayant une caractéristique commune. **HIBISCUS** permet à l'utilisateur de définir n'importe quel type de tâche. La définition de ces types est inclu dans les données. On parlera par exemple du type `nuit` pour désigner l'ensemble des quarts de nuit. Dans les salles d'urgence [20, 7], on a besoin de définir un type `cube`, un type `choc` et un type `suite de soin` (voir sections 6.1.5 et 6.1.6).

2.3 L'horizon de planification

L'horizon de planification est la période du calendrier pour laquelle un horaire de travail doit être conçu. Il est donc caractérisé par une date de début et une date de fin. Chaque jour de l'horizon, plusieurs tâches peuvent être planifiées (remarque : en général, les quarts de nuit sont considérés comme des tâches rattachées au jour précédant la nuit). On appelle *période de l'horizon* tout ensemble de jours non forcément contigus. Une semaine peut désigner selon le contexte une suite de sept jours allant du lundi au lundi ou du dimanche au dimanche.

Les week-ends. Les week-ends sont des périodes particulières débutant à une certaine heure du vendredi ou du samedi et se terminant à une certaine heure du lundi matin. Ces heures peuvent varier selon que le vendredi ou le lundi est férié ou non.

On ne peut donc pas définir un week-end comme une période, mais plutôt comme un ensemble de tâches (toutes celles qui couvrent la période du week-end). On dit qu'un week-end est travaillé par un individu i si et seulement si i effectue l'une des tâches reliées au week-end.

2.4 Les règles

Cette section contient l'ensemble des règles prises en compte complètement ou partiellement par **HIBISCUS**. Ces règles ne sont jamais utilisées toutes en même temps dans un contexte donné, certaines d'entre elles peuvent donc être contradictoires ou légèrement redondantes.

2.4.1 Définitions et notations

Séquences de quarts de même type. Une *séquence de quarts* de type T est une suite de quarts consécutifs de type T telle que le quart précédent et le quart suivant, s'ils existent, ne sont pas de type T.

Les patrons. Un patron est une suite de type de quarts. On différencie les *patrons de jours* sur une période où chaque type du patron correspond à une journée de la période et les *patrons de séquences* qui décrivent des suites d'enchaînements de quarts autorisés. Pour représenter un patron, on écrira les nom des types dont il est composé séparés par le signe /.

2.4.2 Satisfaction de la demande [DEM]

Ces règles assurent que suffisamment de quarts sont pourvus tout au long de la période de planification. Elles visent à obtenir une couverture minimale ou désirée (voir les critères de la section 1.2). Afin de pouvoir exprimer la demande dans le plus grand nombre de contextes, celle-ci peut être définie chaque jour pour un type de quarts T quelconque et pour tout sous-ensemble du personnel. Elle définit une valeur cible ou des bornes sur l'effectif du personnel concerné effectuant un quart de type T.

Exemple 1 *: Les jours de week-ends, l'effectif du personnel infirmier travaillant les quarts de nuit doit être compris entre trois et cinq.*

2.4.3 Les disponibilités [AVA]

Chaque membre du personnel, en fonction de ses qualifications, de son statut régulier ou temps partiel, de la législation en vigueur et de ses choix de vacances, n'est pas disponible en tout temps. Les règles de disponibilité permettent au personnel de définir des horaires flexibles.

2.4.3.1 Pré-affectations et affectations interdites [AVA1]

Certaines portions d'horaires peuvent être prédéterminées (pré-affectations) ou partiellement déterminées (affectations interdites). C'est souvent le cas du personnel à temps partiel qui travaille souvent dans plusieurs institutions. Les jours de vacances sont aussi traités comme des pré-affectations dans la plupart des contextes.

Exemple 2 *: Le médecin X n'est disponible que pendant les deux premières semaines de l'horizon.*

Exemple 3 *: Le médecin Y n'est disponible que pour les quarts de jour.*

2.4.3.2 Propositions pour les vacances [AVA2]

Dans le cas où les vacances ne sont pas des pré-affectations, le personnel peut spécifier un ensemble de jours candidats ainsi qu'un nombre minimum et un nombre maximum de jours de vacances à choisir parmi ces jours candidats.

Exemple 4 *: L'infirmière Z doit prendre entre cinq et dix jours de vacances au cours des deux dernières semaines de l'horizon.*

2.4.4 Respect de la charge de travail [WOR]

Suivant les contextes, la charge spécifiée dans les contrats de travail peut s'exprimer en heures de travail ou en nombre de quarts. Dans tous les cas, cette règle impose un minimum et un maximum à la charge réelle. Il est aussi possible de répartir la charge quasi-uniformément ou selon une distribution voulue en définissant des charges sur des sous-périodes de l'horizon.

Exemple 5 *: Le médecin X doit faire cinq quarts D, cinq quarts E et trois quarts N au cours l'horizon.*

Exemple 6 *: L'infirmière Y doit travailler exactement 80 heures toutes les deux semaines.*

2.4.5 Règles ergonomiques [ERG]

C'est la catégorie de règles la plus fournie et la plus hétérogène. Elle contient toute règle visant à maintenir un niveau de qualité pour les horaires individuels (voir les critères de Warner à la section 1.2). Certaines d'entre elles peuvent être obligatoires, d'autres, laissées au choix de chacun des membres du personnel.

2.4.5.1 Patrons de jours sur une période donnée [ERG1]

Certaines périodes particulières de l'horizon (les week-ends, les jours fériés) doivent observer certains patrons. On dit qu'une période observe un patron (les deux ayant obligatoirement la même taille) si et seulement si chaque jour de la période exhibe un quart du type indiqué à la même position dans le patron. La figure 2.1 présente un horaire dans lequel les périodes grisées satisfont le patron `D/E/N`.

D			D	E	N		D	D	D		D	E	N		

Figure 2.1 – Exemple d'horaire satisfaisant le patron `D/E/N` sur les périodes grisées.

Exemple 7 *: Le personnel préfère ne pas avoir à travailler un seul des deux jours du week-end (week-end brisé).*

2.4.5.2 Patrons de séquences [ERG2]

L'enchaînement de certaines activités est interdit ou incommode pour le personnel. Les patrons de séquences décrivent les successions possibles des activités sur l'horizon. En pratique, on utilise surtout des patrons de séquences binaires et ternaires. Soit $\mathcal{T} = \{T_1, \ldots, T_n\}$ un ensemble de n types de quart formant une partition de l'ensemble des quarts. Un horaire quelconque peut alors être divisé en une succession $(s_1, \ldots, s_m)$ de séquences de types de $\mathcal{T}$. Cet horaire est valide par rapport à un ensemble $\mathcal{P}_\ell$ de patrons de longueur ℓ autorisés si et seulement si pour toute sous-séquence de $(s_1, \ldots, s_m)$ de taille ℓ, il existe un patron de $\mathcal{P}_\ell$ correspondant à cette sous-séquence (c'est-à-dire pour lequel chaque type correspond dans l'ordre au type de celle-ci). Par

exemple, étant donné un ensemble de patrons ternaires autorisés incluant $T_2/T_5/T_3$ et $T_5/T_3/T_1$, l'horaire de la figure 2.2 est valide.

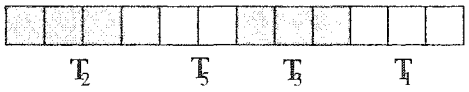


Figure 2.2 – Horaire découpé en séquences de types.

Exemple 8 *: Le médecin X préfère effectuer des blocs de travail homogènes à un type* `jour`, `soir` *ou* `nuit`, *tout en respectant la règle du cycle circadien (la rotation* `jour`/`soir`/`nuit`*). Les patrons autorisés sont donc* `jour / congé / jour`, `jour / congé / soir`, `soir / congé / soir`, `soir / congé / nuit`, `nuit / congé / nuit` *et* `nuit / congé / jour`.

2.4.5.3 Nombre d'affectations consécutives [ERG3]

La longueur des séquences des activités d'un certain type peut être bornée.

Exemple 9 *: Le médecin X, lorsqu'il travaille de nuit, tient à travailler trois nuits d'affilée.*

Exemple 10 *: Une infirmière ne doit pas travailler plus de sept jours consécutifs.*

2.4.5.4 Week-ends consécutifs [ERG4]

Les week-ends font souvent l'objet de règles de distribution spéciales. Ces règles spécifient un minimum et un maximum pour la longueur des séquences de week-ends travaillés et non travaillés de chaque membre du personnel.

Exemple 11 *: L'alternance des week-ends travaillés et non travaillés doit être respectée.*

Exemple 12 *: Le médecin X ne doit pas travailler au cours de deux week-ends consécutifs.*

2.4.5.5 Séquences consécutives [ERG5]

Une séquence vérifiant certaines propriétés (taille, patrons, etc...) doit être immédiatement suivie par une autre séquence vérifiant d'autres propriétés.

Exemple 13 *: Après trois nuits, un médecin doit avoir un minimum de trois jours de congé.*

2.4.5.6 Distribution des quarts importuns [ERG6]

Cette règle assure que pour chacun des membres du personnel, le nombre de quarts importuns n'est pas trop éloigné du quota en le confinant entre deux bornes.

Exemple 14 *: Un médecin ne doit pas assister à plus d'une réunion au cours de l'horizon.*

Exemple 15 *: Un médecin ne doit pas effectuer plus de cinq quarts de week-end par horizon.*

2.4.5.7 Distribution entre plusieurs types de quarts [ERG7]

Le personnel peut spécifier une distribution souhaitable entre plusieurs types de quarts $(T_k)_{1 \leq k \leq n}$ disjoints. Dans le cas où elle est souple, cette règle vise à équilibrer les écarts en valeur absolue par rapport aux proportions de chaque type de quart.

Exemple 16 *: Le rapport entre le nombre de quarts* `cubes` *et le nombre de quarts* `chocs` *des horaires individuels doit se rapprocher le plus possible du rapport entre le nombre total de quarts* `cubes` *et le nombre total de quarts* `chocs` *disponibles au cours de la période de planification.*

Exemple 17 *: L'infirmière Y doit effectuer au cours de l'horizon deux fois plus de quarts de* `soir` *que de quarts de* `nuit`.

2.4.5.8 Préférences et aversions [ERG8]

Le personnel peut spécifier individuellement ses préférences et ses aversions en termes d'affectations désirables, neutres et non désirables. Cette règle est fondamentalement souple. On cherchera à maximiser les préférences et à minimiser les aversions.

2.4.6 Règles d'équité [FAI]

Les règles d'équité servent à garantir à ce qu'aucun membre du personnel ne se trouve avantagé par rapport aux autres relativement à la satisfaction d'un critère souple. Il peut s'agir des préférences et aversions ([ERG8]), ou de n'importe qu'elle autre règle souple.

2.5 Particularité des unités de soins infirmiers

Cette section répertorie quelques cas particuliers rencontrés au MUHC (Mc Gill Universty Health Center) et pris en compte dans le modèle du logiciel IRIS auquel nous tenterons de nous comparer.

2.5.1 Définition de la demande

Dans ce contexte, une journée est divisée en *périodes de quota* (intervalles de temps). Étant donné un ensemble de quarts, les périodes sont choisies maximales de telle sorte que l'ensemble des quarts simultanés ne change pas au cours d'une période. De ce fait, une période de quota est caractérisée par l'ensemble des quarts qui la couvrent. La figure 2.3 montre le découpage d'une journée en période de quota.

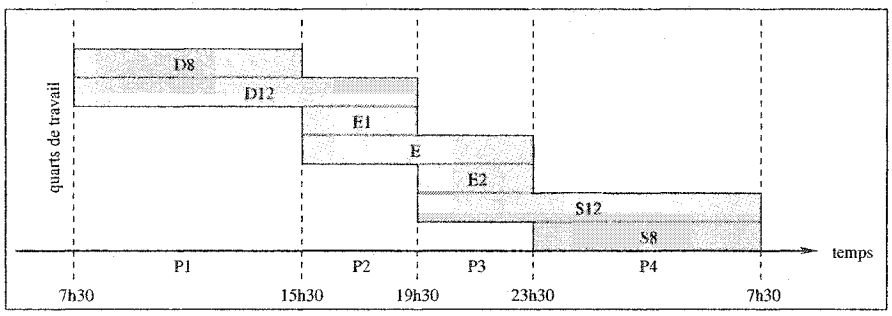


Figure 2.3 – Exemple de découpage d'une journée en périodes de quota : P_1, P_2, P_3 et P_4.

La demande est exprimée pour chacune des périodes de quota de la journée et pour le personnel d'un niveau de compétence particulier. Reprenons l'exemple de la figure 2.3. Une contrainte de satisfaction de la demande pourrait être formulée ainsi :

Exemple 18 *: l'effectif du personnel de niveau de compétence 1 travaillant pendant la période P_3 doit être compris entre trois et cinq.*

2.5.2 Vacances et week-ends

Dans ce contexte, les vacances sont des congés payés (un jour de vacances compte pour une charge de 8 heures) et aucun congé payé ne peut être attribué un jour de week-end. De plus, les jours de week-ends situés au milieu d'une période de vacances ne peuvent être travaillés.

2.5.2.1 Exception pour la règle des week-ends non brisés

Les week-ends doivent, à quelques exceptions près, être non brisés. Cette règle est appliquée à tous les week-ends de l'horizon excepté ceux qui délimitent des périodes de vacances. Le tableau 2.1 répertorie les patrons `congé/travail` possibles en fonction de la fin de la semaine précédente et le début de la semaine suivante. On parle d'une fin (resp. d'un début) de semaine de `travail` si le dernier (resp. premier) jour de la semaine différent d'un congé est un jour travaillé. On parle d'une fin ou d'un début de semaine de `vacances` si ce jour est un jour de vacances.

Tableau 2.1 – Week-ends brisés et non brisés autorisés

Fin de la semaine précédente	Week-end		Début de la semaine suivante
	Samedi	Dimanche	
vacances	congé	travail	travail
vacances	congé	congé	travail
travail	congé	congé	vacances
travail	travail	congé	vacances
travail	congé	congé	travail
travail	travail	travail	travail
vacances	congé	congé	vacances

2.5.2.2 Exception pour la règle sur les séquences de week-ends

On retrouve la règle sur les séquences de week-ends travaillés et non travaillés mais avec une exception pour les week-ends appartenant à une période de vacances. Une période de vacances caractérise un ensemble de jours consécutifs de congés et de vacances. Un week-end compris entre un début de semaine de vacances et une fin de semaine de vacances ne doit pas être travaillé. Après la période de vacances cependant, le rythme des séquences doit reprendre comme s'il n'avait pas été interrompu. La figure 2.4 est un exemple d'alternance de week-ends travaillés et non travaillés valide en période de vacances.

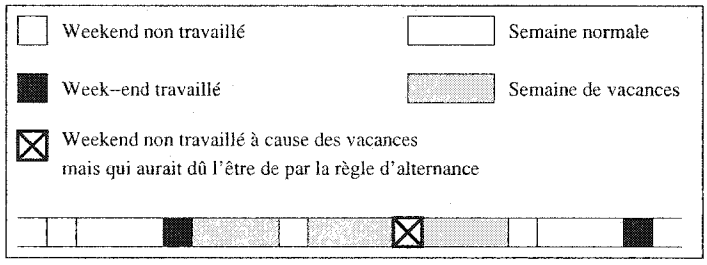


Figure 2.4 – Conséquences des vacances sur l'enchaînement des week-ends

CHAPITRE 3

LA PROGRAMMATION PAR CONTRAINTES (PC)

La programmation par contraintes (PC) est un paradigme de programmation qui permet de dissocier la représentation du problème de sa résolution. En PC, on ne décrit pas l'algorithme qui permet de trouver une solution mais plutôt les solutions elles-mêmes, en exprimant les propriétés et les relations entre les variables du problème. Une fois le problème défini, on utilise un algorithme appelé *solveur de contraintes* pour le résoudre.

Un *solveur de contraintes* est spécifique à un ensemble de contraintes définies sur un domaine particulier que l'on appelle *système de contraintes.* Il existe des solveurs pour des *systèmes de contraintes* aussi divers que les contraintes sur les arbres finis, les contraintes linéaires de type égalité sur les réels, les contraintes linéaires sur les réels, les contraintes non-linéaires sur les réels, les contraintes sur les variables de domaine fini, etc... Pour chacun d'entre eux, le *solveur de contraintes* utilise des algorithmes différents. Le système de contraintes sur les variables de domaines finis est celui qui donne lieu au plus grand nombre d'applications pratiques. C'est d'ailleurs celui que nous utilisons pour modéliser notre problème de confection d'horaires. Les problèmes exprimés sur les domaines finis s'appellent aussi CSP (*Constraint Satisfaction Problem*). Ce sont des problèmes combinatoires souvent NP difficiles.

Le paradigme de programmation par contraintes est orthogonal aux autres paradigmes de programmation que sont la programmation impérative, la programmation fonctionnelle, la programmation logique ou la programmation orientée objet : il existe des langages de programmation par contraintes logiques (comme CHIP, Eclipse,

PrologV ou Gnu-Prolog), fonctionnels (comme Oz), impératifs (comme CHOCO) ou orientés objet (comme les bibliothèques C++ et Java de Ilog Solver).

3.1 Les CSP

Les CSP ou réseaux de contraintes [34] ont déjà été largement étudiés depuis plusieurs années dans le cadre de recherches en intelligence artificielle. Ils sont définis comme suit :

Définition 3.1 (CSP) *Un CSP est un triplet* $\langle X, \mathcal{D}, \mathcal{C} \rangle$ *où*

- X *est un ensemble de variables de domaine fini,*
- $\mathcal{D}$ *est l'ensemble des domaines des variables, et*
- $\mathcal{C}$ *est un ensemble de contraintes qui restreignent les valeurs que les variables peuvent prendre simultanément.*

Vocabulaire et notations

Soit $X = \{x_1, ..., x_n\}$ un ensemble de variables de domaines finis. On note le domaine initial d'une variable x_i par $D^\circ_{x_i}$. On représente par D°_X l'union des domaines des variables dans X. On appelle espace de recherche le produit cartésien des domaines initiaux ($S = D^\circ_{x_1} \times ... \times D^\circ_{x_n}$). Une instanciation partielle $\mathcal{I}$ des variables de X est une application qui associe à chaque variable $x \in X$ un domaine courant $D_x \subset D^\circ_x$. Dans le cadre de ce mémoire, $\mathcal{I}$ désignera l'ensemble des domaines courants de l'instanciation partielle. Si les domaines courants sont réduits à un élément, on parle alors d'une affectation. Dans le cadre de ce mémoire, une affectation t sera le tuple des valeurs affectées,*i.e.* un élément de S. On utilisera la notation $t[x]$ pour désigner la valeur de la variable x dans l'affectation t. À toute instanciation partielle peut être associée un sous-ensemble de l'espace de recherche $S_\mathcal{I} = \{t \in S \mid \forall x \in X,\ t[x] \in D_x\}$. Soient $\mathcal{I}$ et $\mathcal{I}'$ deux instanciations partielles. On dit que $\mathcal{I}'$ étend $\mathcal{I}$ ssi $S_{\mathcal{I}'} \subset S_\mathcal{I}$. Soit Y un sous-ensemble de X, on note $t_{|_Y}$ la projection de l'affectation t sur Y. Soit C une contrainte sur X. On note V_C le sous-ensemble des variables liées par C et T_C l'ensemble des affectations de V_C qui vérifient la contrainte. Le cardinal de l'ensemble V_C, *i.e.* le nombre de variables liées par la contrainte C, est aussi appelé *l'arité* de la contrainte C. On utilisera le symbole $\bigotimes$ pour noter un produit cartésien à n dimension ($\bigotimes_{1 \leq i \leq n} E_i \Leftrightarrow E_1 \times ... \times E_n$).

Une solution d'un CSP est une affectation t de X qui satisfait toutes les contraintes, *i.e.* t est une solution du CSP $\langle X, \mathcal{D}, \mathcal{C} \rangle$ si et seulement si $\forall C \in \mathcal{C},\ t_{|_{V_C}} \in T_C$. Suivant le problème, on peut vouloir trouver :

- juste une solution (la première),
- toutes les solutions,
- une solution optimale, ou seulement une *bonne* solution, compte tenu d'une fonction d'évaluation définie sur tout ou une partie des variables.

Le domaine de la programmation mathématique auquel les CSP s'apparentent le plus est la programmation linéaire en nombres entiers. Le formalisme de CSP est cependant plus général car il permet de prendre en compte des contraintes plus complexes. Modéliser un problème équivalent sous la forme d'un programme linéaire en nombres entiers nécessite de faire des approximations qui complexifient parfois beaucoup le modèle.

3.2 Résolution des CSP

Il existe deux grandes familles de méthodes pour résoudre un CSP : les méthodes qui parcourent explicitement l'espace des solutions et celles qui l'explorent implicitement.

Dans le premier cas, l'algorithme part d'une solution initiale quelconque (réalisable ou non) et se déplace de solution en solution jusqu'à trouver une solution réalisable ou la meilleure solution. L'algorithme *générer puis tester* est le plus naïf de cette famille. Il consiste à générer des affectations dans un ordre quelconque et de tester ensuite leur cohérence. En pratique, on ne peut pas parcourir l'ensemble de l'espace des solutions possibles en un temps raisonnable. C'est pourquoi la plupart des méthodes sont heuristiques et se cantonnent à des sous espaces (voisinages) de solutions. Parmi ces méthodes, on trouve les heuristiques de recherche locale et des méta-heuristiques telles que la recherche tabou.

Dans le second cas, l'algorithme part d'une affectation vide et la construit au fur et à mesure. Dans ce cas, l'algorithme parcourt des solutions partiellement instanciées, qui représentent chacune un ensemble de solutions, tout en essayant de maintenir un certain niveau de cohérence. L'intérêt de cette méthode par rapport aux précédentes est la possibilité d'éliminer d'un seul coup un sous-ensemble important de solutions

non valides et par la même occasion de parcourir implicitement l'espace des solutions dans sa globalité. De tels algorithmes permettent donc de trouver une solution s'il en existe une, et dans le cas contraire, de prouver qu'il n'en existe pas. On les appelle *algorithmes de recherche systématique.*

Algorithmes de recherche systématique

Les algorithmes de recherche systématique parcourent l'espace des solutions en le divisant successivement en sous ensembles de plus en plus raffinés. La procédure est une fouille arborescente dans laquelle chaque noeud représente un sous ensemble particulier de solutions. S'il n'est pas cohérent, *i.e.* s'il ne contient pas de solution au problème, on l'élimine et on continue la recherche sur un autre sous-ensemble que l'on avait préalablement laissé en suspens (retour en arrière). Dans le cas contraire, on subdivise le sous espace en sous espaces fils, et on applique récursivement la procédure à l'un d'entre eux tout en laissant les autres en suspens. Ce mécanisme de retour en arrière est aussi appelé *backtracking.*

```
Fonction BT(I : Instanciation partielle) : retourne booléen
si ¬ cohérence(I) alors
  retourner faux ; {Retour arrière}
fin si
si solution(I) alors
  retourner vrai ; {On a trouvé une solution}
fin si
⟨I_1, ..., I_n⟩ ← division(I) ;
resultat ← faux ;
i ← 0 ;
tant que ¬resultat ∨ i ≤ n faire
  resultat ← BT(I_ordre_I(i)) ;
  i ← i + 1 ;
fin tant que
retourner resultat ;
```

Algorithme 3.1: Recherche systématique avec retour en arrière (*backtracking*).

L'algorithme général de telles procédures se décrit simplement sous forme récursive (voir algorithme 3.1). La fonction cohérence teste la cohérence de l'instanciation partielle $\mathcal{I}$. Éventuellement, dans des formes avancées de l'algorithme, cette fonction peut éliminer des valeurs des domaines des variables de X qui ne sont pas encore liées. On

appelle cela du *filtrage*. La fonction cohérence en elle-même représente l'étape de *propagation des contraintes*. La fonction solution teste si l'instanciation partielle $\mathcal{I}$ est une affectation, *i.e.* si les domaines courants des variables sont réduits à un élément. La fonction division associe à $\mathcal{I}$ un ensemble d'instanciations partielles étendues. On appelle cette étape le *branchement*. Finalement, la fonction $ordre_{\mathcal{I}}$ est une bijection de $[1..n]$ qui indique dans quel ordre les sous espaces sont parcourus.

3.2.1 Cas particulier de l'optimisation

Dans beaucoup d'applications, il ne suffit pas de trouver une solution, il faut que ce soit une bonne ou parfois même la meilleure solution. La qualité des solutions est alors mesurée par une fonction appelée *fonction objectif*, qui associe à chaque solution une valeur numérique. Le problème consiste à trouver une solution qui satisfasse les contraintes et qui minimise (ou maximise) la fonction objectif. De tels problèmes sont aussi appelés CSOP *(Constraint Satisfaction Optimization Problem)*.

L'algorithme le plus utilisé pour résoudre les CSOP est le *branch-and-bound*. Il nécessite une fonction qui associe à chaque instanciation partielle $\mathcal{I}$ un minorant (dans le cas d'un problème de minimisation) de l'ensemble de valeurs de la fonction objectif pour les solutions qui étendent $\mathcal{I}$. Notons f la fonction objectif et h la fonction minorante. h doit vérifier la propriété suivante :

"Pour toute instanciation partielle $\mathcal{I}$, $\forall t \in S_{\mathcal{I}}$, $h(\mathcal{I}) \leq f(t)$."

Le *branch-and-bound* de la PC (voir algorithme 3.2) ne diffère pas beaucoup du *backtracking* : c'est une fouille arborescente de l'espace des solutions en profondeur d'abord. Cependant, cette fouille est complète et dispose, grâce à la fonction minorante, d'un moyen supplémentaire pour couper des branches de l'arbre : si la valeur de l'heuristique sur un instanciation partielle $\mathcal{I}$ est supérieure à la valeur de la meilleure solution trouvée, cela signifie que l'on ne pourra pas trouver de solution meilleure parmi celle qui étendent $\mathcal{I}$.

L'efficacité de l'algorithme repose à la fois sur la qualité de la fonction minorante comme approximation de la fonction objectif et sur la rapidité avec laquelle est trouvée une bonne première solution (qualité de la stratégie de recherche). Lorsque l'on n'a pas besoin de la solution optimale, la recherche de solution peut toujours être arrêté lorsqu'une bonne solution a été trouvée. De plus, si l'on initialise la variable *bound* de l'algorithme avec un bon majorant du coût optimal plutôt que $+\infty$, la recherche

```
Fonction BB(I : Instanciation partielle, bound : Réel) : retourne booléen
si ¬ cohérence(I) ∨ h(I) ≥ bound alors
  retourner faux ; {Retour en arrière}
fin si
si solution(I) alors
  si f(I) < bound alors
    bound ← f(I) ;
  fin si
  retourner vrai ; {On a trouvé une solution}
fin si
⟨I_1, ..., I_n⟩ ← division(I) ;
resultat ← faux ;
pour i = 1..n faire
  resultat ← resultat ∨ BB(I_ordre_I(i), bound) ;
fin pour
retourner resultat ;
```

Algorithme 3.2: Branch-and-bound pour les CSOP.

de l'optimum est accélérée car l'algorithme coupe un plus grand nombre de branches inutiles.

3.2.2 Branchement

Le branchement est l'opération qui divise l'espace de recherche en sous espaces. Généralement binaire, il n'est pas forcement équilibré, *i.e.* les sous espaces obtenus ne sont pas obligatoirement tous de la même taille. Cependant, ils doivent former une partition de l'espace courant, sinon la recherche est soit redondante, soit incomplète.

Définition 3.2 (arbre de recherche) *On appelle* arbre de recherche *ou* arbre des branchements *l'arbre généré par l'application de l'algorithme 3.1 sur un CSP. C'est un arbre dont les noeuds représentent des solutions instanciations et dont les feuilles correspondent soit à une solution réalisable, soit à un sous ensemble de solutions non valides (au sens de la fonction cohérence). L'application de l'algorithme 3.1 sur le CSP défini par* $\langle \{x,\ y\},\ \{\{0,1\}_x,\ \{0,1\}_y\},\ \{x \neq 0,\ x+y \leq 1\}\rangle$ *pourrait donner l'arbre de recherche représenté à la figure 3.1.*

Le branchement que l'on appellera par la suite *branchement classique*, consiste à choisir une variable future x (*i.e.* une variable non encore instanciée) et à instancier successivement x avec les valeurs de son domaine. C'est le choix qui a été fait pour

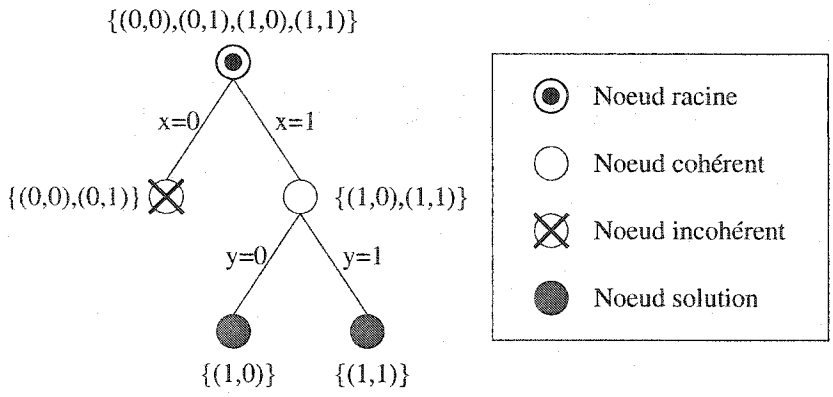


Figure 3.1 – Exemple d'arbre de recherche.

générer l'arbre de la figure 3.1. Une alternative consiste à choisir également une valeur $v \in D_x$ et ne considérer que les deux branches $x = v$ et $x \neq v$. Il faut noter toutefois que pour utiliser ces techniques de branchement en pratique, il est nécessaire de spécifier une façon de choisir les variables et les valeurs concernées par le branchement. Nous parlerons de l'impact de ces choix sur la recherche dans la section 3.2.4.

3.2.3 Algorithmes de filtrage et propagation des contraintes

En PC, chaque contrainte est une entité propre qui joue un rôle particulier lors de la résolution par l'intermédiaire d'un algorithme de filtrage. Un algorithme de filtrage sert à maintenir un certain niveau de cohérence entre les domaines des variables de la contrainte. À chaque noeud de l'arbre de recherche, les algorithmes de filtrage des contraintes sont appliqués au CSP afin de détecter au plus vite son incohérence, ou de le réduire en filtrant des valeurs dans les domaines pour faciliter la poursuite de la recherche. La propagation des contraintes n'assure qu'une cohérence partielle, *i.e.* on ne peut généralement pas savoir si le CSP résultant a une solution.

3.2.3.1 Les propriétés de cohérence partielle

Plusieurs propriétés de cohérence partielle ont été définies dans la littérature. Le cas particulier des CSP binaires (CSP dont les contraintes lient au plus deux variables) a été largement étudié du fait qu'il est possible de transformer un CSP quelconque en un CSP binaire. Cependant, dans la pratique, les solveurs travaillent, le plus souvent, directement sur des CSP non binaires.

La propriété la plus simple est la cohérence de noeud qui caractérise les CSP dont les variables vérifient les contraintes unaires. Cette propriété s'obtient simplement en éliminant des domaines les valeurs incohérentes avec les contraintes unaires. La propriété la plus populaire est la cohérence d'arc car elle permet d'effectuer plus de filtrages que la cohérence de noeud avec un coût en calcul supplémentaire négligeable. Elle peut en outre se généraliser au cas des contraintes non binaires.

Définition 3.3 (Cohérence d'arc (généralisée)) *Soit un CSP* $\mathcal{P} = \langle X, \mathcal{D}, \mathcal{C} \rangle$. *Étant donnés une contrainte* $C \in \mathcal{C}$, *une variable* $x \in V_C$, *on dit qu'une valeur* $d \in D_x$ *a un support dans* C *ssi il existe un tuple dans* T_C *tel que* $t[x] = d$. *La contrainte* C *vérifie la cohérence d'arc ssi chaque valeur* d *de chaque variable* $x \in V_C$ *a un support dans* C. *Le CSP* $\mathcal{P}$ *vérifie la cohérence d'arc ssi toutes ses contraintes la vérifient et si de plus, il vérifie la cohérence de noeud.*

D'autres propriétés de cohérence partielles ont été établies par la suite, notamment la k-cohérence et la k-cohérence forte qui généralise la cohérence de noeud et la cohérence d'arc. Mais les algorithmes permettant de maintenir la k-cohérence d'un CSP ont une complexité exponentielle. Ainsi, en pratique, on se cantonne à la k-cohérence pour $k \leq 2$, c'est à dire la cohérence d'arc. La figure 3.2 donne l'exemple d'un CSP qui vérifie la cohérence d'arc et qui pourtant n'a pas de solution.

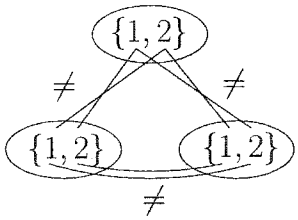


Figure 3.2 – Exemple de CSP vérifiant la cohérence d'arc.

3.2.3.2 Algorithmes de propagation

Nous décrivons dans cette section quelques implantations possibles de la fonction cohérence de l'algorithme 3.1. Un algorithme de propagation combine les différents algorithmes de filtrage des contraintes afin d'effectuer un compromis entre la quantité et la complexité du filtrage.

L'algorithme de *backtracking* le plus simple possède une phase de propagation dans laquelle il applique la cohérence d'arc sur les contraintes dont toutes les variables sont instanciées. Cela permet de détecter les incohérences dès qu'elles apparaissent et donne ainsi un net avantage par rapport à l'algorithme *generate and test.*

La technique du *forward checking* est la façon la plus simple de prévoir les incohérences futures. Elle consiste à appliquer la cohérence d'arc aux contraintes qui portent sur la variable courante et au moins une variable future et élimine ainsi toute valeur du domaine d'une variable future qui est en conflit avec l'affectation courante. Si au cours de la procédure un domaine devient vide, cela signifie que la solution partielle courante est incohérente. Notons qu'il n'est alors plus nécessaire de tester la cohérence entre l'instanciation courante et les variables passées car le *forward checking* assure que toutes les valeurs restantes d'un domaines le sont.

La technique appelée *looking ahead* ou *Maintaining Arc Consistency (MAC)* va plus loin que le *forward checking* en appliquant également la cohérence d'arc sur les contraintes qui ne lient que des variables futures. Elle assure ainsi que le CSP résultant vérifie la propriété de cohérence d'arc. Cette procédure effectue plus de filtrage que le *forward checking* mais demande cependant plus de travail à chaque noeud de l'arbre de recherche.

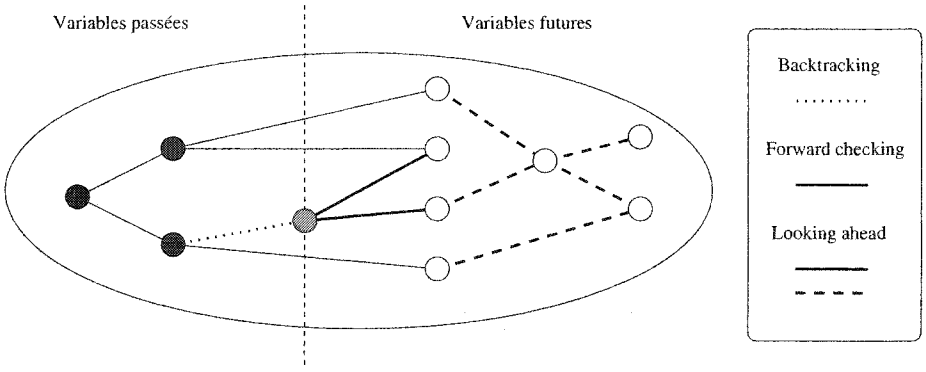


Figure 3.3 – Révision des arcs pour plusieurs algorithmes de propagation.

La figure 3.3 illustre les différences entre les trois techniques énoncées précédemment dans le cas d'un CSP binaire (les variables sont représentées par les noeuds et les contraintes par les arcs). Le *forward checking* et le *looking ahead* sont les deux techniques de propagation les plus utilisées pour les CSP binaires car la cohérence d'arc est peu coûteuse dans ce cas. Lorsqu'un CSP comporte des contraintes qui lient

un grand nombre de variables il n'est pas judicieux d'utiliser un filtrage par cohérence d'arc car sa complexité est exponentielle en le nombre de variables.

3.2.3.3 Cas des contraintes non binaires

Pour les contraintes non binaires d'arité faible ou dont la représentation en extension est de taille raisonnable, on peut encore utiliser un algorithme de filtrage basé sur la cohérence d'arc généralisée. Dans le cas contraire, il y a plusieurs façons de simplifier le problème :

- transformer la contrainte en une conjonction de contraintes unaires et binaires équivalente et appliquer à chacune la cohérence de noeud ou d'arc,
- utiliser un algorithme de filtrage basé sur une propriété de cohérence plus faible comme la cohérence de bornes quand les domaines sont des sous-ensembles de $\mathbb{Z}$,
- utiliser la sémantique de la contrainte pour élaborer un algorithme de filtrage de complexité polynomiale qui maintient la cohérence globale (au niveau de la contrainte).

Dans le premier cas, on ajoute beaucoup à la compléxité du modèle, dans le second, on perd en quantité de filtrage, et le troisième cas n'est pas toujours possible. Actuellement, la recherche d'algorithmes de filtrage performants est très active [18, 29, 45].

Cohérence de bornes

Elle s'applique aux contraintes $>$, $\geq$, $=$, $\neq$ qui comparent des expressions algébriques liant plusieurs variables. Pour l'établir, on ne raisonne que sur les valeurs minimum et maximum des domaines et sur les domaines étendus $D_x^\star = [D_x^{min}, D_x^{max}]$. Se référer à [44] pour plus d'informations.

Définition 3.4 (Cohérence de bornes) *Une contrainte C vérifie la cohérence de bornes ssi pour chaque variable $x \in V_C$ et pour chaque valeur $d \in \{D_x^{min}, D_x^{max}\}$, il existe un tuple $t \in \bigotimes_{y \in V_C \setminus \{x\}} D_y^\star$ tel que $C(t, d)$ est vérifiée.*

Contraintes globales

On appelle *contrainte globale* toute contrainte d'arité "élevée" pour laquelle il existe un algorithme de filtrage performant (polynomial) qui maintient un "haut" niveau de cohérence entre les variables de la contrainte.

Dans certains cas, une contrainte globale remplace un ensemble de contraintes de plus bas niveau sur lequel l'application de la cohérence d'arc n'est pas assez efficace. C'est la cas de la contrainte AllDifferent(X) [39] qui assure que les variables de X aient toutes une valeur différente. En utilisant un algorithme polynomial (recherche du couplage maximal dans un graphe biparti), elle effectue plus de filtrage que l'application de la cohérence d'arc sur chacune des contraintes binaires $x \neq y$, x et y étant deux variables distinctes de X.

Plusieurs contraintes globales ont été définies pour répondre à des problèmes rencontrés dans des champs particuliers d'application. La contrainte Stretch [36] est très utile pour les problèmes de confection d'horaires. D'autres contraintes [5] existent pour les problèmes d'ordonnancement, de placement dans l'espace ou de tournées de véhicules.

3.2.4 Stratégies et heuristiques de recherche

On appelle *stratégie de recherche* l'ensemble des procédures de choix utilisées lors du branchement (choix de premier niveau) et de la fonction $ordre_{\mathcal{I}}$ (choix de second niveau). Si on considère le branchement classique, le choix de premier niveau correspond à ce que l'on appelle communément le choix de variable et le choix de second niveau au choix de valeur. Mais pour d'autres choix de branchement, cette séparation n'existe plus. Plus généralement, les choix de premier niveau vont directement influencer la structure de l'arbre de recherche (nombre de noeuds explorés, profondeur moyenne) alors que ceux du second niveau vont seulement modifier l'ordre dans lequel l'algorithme va trouver les solutions (et les mauvaises branches). Pour trouver rapidement une bonne première solution, il est important de considérer les deux.

Ces choix peuvent être statiques, lorsque l'ordre est défini à l'avance, ou dynamiques, lorsque l'ordre est déterminé au cours de la recherche. Dans ce dernier cas, le choix est plus pertinent car il peut tenir compte de l'état du système, mais il nécessite des calculs supplémentaires à chaque noeud de l'arbre. Le principe du *first fail* est souvent utilisé pour les choix de premier niveau. Lorsqu'il s'agit d'un choix de variable, il se traduit par : choisir la variable de plus petit domaine d'abord. Pour le choix de second niveau cependant, on applique souvent le principe du *succeed first* en choisissant la valeur qui, après instanciation, maximise le produit de la taille des domaines après filtrage.

Il n'existe aucun résultat théorique sur les stratégies de recherche, seulement quelques règles empiriques telles que le *first fail* ou le *succeed first*. Dans tous les cas, la stratégie à utiliser dépend fortement de la structure du problème étudié.

3.2.4.1 Représentation des stratégies

Pour représenter les stratégie de façon compacte, on peut utiliser le formalisme des graphes ET/OU. Donnons tout d'abord quelques définitions

Définition 3.5 (graphe ET/OU (voir figure 3.4)) *Un graphe ET/OU $\mathcal{G}$ est un graphe orienté dont chaque noeud représente un sous-problème. Les successeurs d'un noeud représentent les sous-problèmes en lequels le problème associé au noeud père peut être décomposé. $\mathcal{G}$ contient un noeud spécial, le noeud racine, qui représente le problème initial à résoudre. $\mathcal{G}$ contient aussi un ensemble de noeuds, dénommés feuilles, qui sont de deux types : les noeuds terminaux qui représentent les problèmes solubles et les noeuds non-terminaux qui représentent les problèmes incohérents. Chaque noeud est soit un noeud ET, soit un noeud OU. Un noeud OU est résolu ssi l'un de ses fils est résolu tandis qu'un noeud ET est résolu ssi tous ses fils sont résolus. Sans perte de généralité, on considère que les feuilles sont des noeuds OU.*

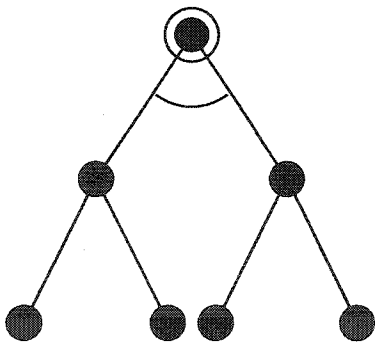


Figure 3.4 – Exemple de graphe ET/OU.

Définition 3.6 (graphe solution (voir figure 3.5)) *Un graphe solution $\mathcal{S}$ est un sous-graphe de $\mathcal{G}$ tel que :*

- *$\mathcal{S}$ contient le noeud racine,*
- *si $\mathcal{S}$ contient un noeud ET, alors il contient tous ses fils, et*
- *si $\mathcal{S}$ contient un noeud OU, alors il contient uniquement l'un de ses fils.*

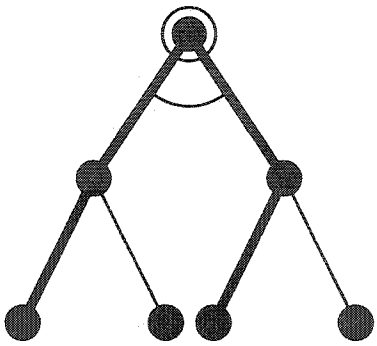


Figure 3.5 – Graphe solution du graphe ET/OU de la figure 3.4.

Trouver une solution au problème initial, c'est trouver un graphe solution dont tous les noeuds sont résolus, donc, en particulier, le noeud de départ. Graphiquement, on représente les noeuds ET et les noeuds OU comme sur les figures 3.6(a), 3.6(b), 3.6(c) et 3.6(d).

La figure 3.7 représente un modèle simple de stratégie au formalisme des graphes ET/OU. Ce graphe indique simplement que l'on essaie de résoudre successivement et dans l'ordre *ordre* les noeuds $x \leftarrow d?$ qui représentent les sous-problèmes d'affectation d'une valeur à une variable. La recherche est complète et non redondante car tous les noeuds $x \leftarrow d?$ sont visités une seule et unique fois. On voit apparaître clairement sur ce graphe les deux niveaux de choix que nous mentionnons plus haut. Le nombre de niveaux de choix dépend directement du nombre de niveaux de décomposition du problème initial. Dans les stratégies que nous décrivons dans le chapitre 5, nous utilisons plus de deux niveaux.

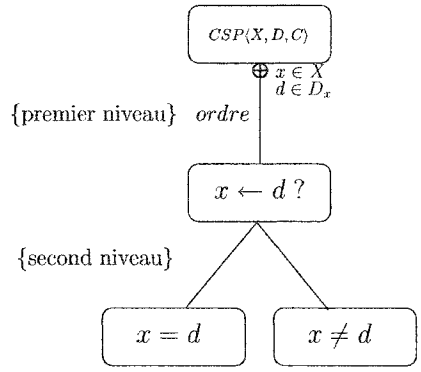
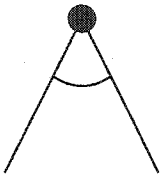
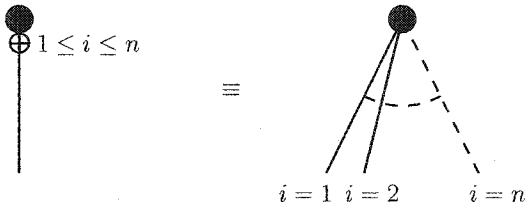


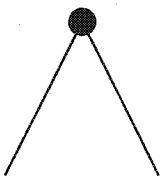
Figure 3.7 – Description générale d'une stratégie.



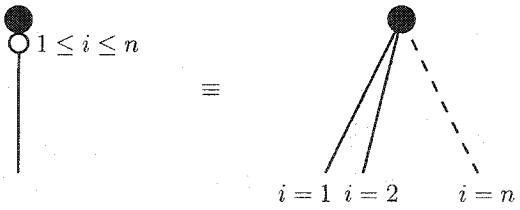
(a) Noeud ET simple.



(b) Noeud ET multiple.



(c) Noeud OU simple.



(d) Noeud OU multiple.

Figure 3.6 – Représentations graphiques des noeuds ET et OU.

Cet exemple est un peu simpliste, mais l'utilisation du formalisme ET/OU devient intéressante pour faire apparaître certaines décompositions particulières comme nous le voyons dans la section suivante.

3.2.4.2 Différentes décompositions utilisées

Un CSP peut être décomposé de deux façons : on parle de décomposition par les variables et de décomposition par les valeurs. Soient $\langle X_1, \ldots, X_n \rangle$ une partition de l'ensemble des variables et $\langle D_1, \ldots, D_m \rangle$ une partition de l'ensemble des valeurs. Dans le premier cas, on va considérer les sous-problèmes `Affectation`(X_i) qui consistent à affecter toutes les variables du sous-ensemble X_i (voir figure 3.8) alors que dans le second cas, on va considérer les sous-problèmes `Affectation`(D_i) qui consistent à affecter à l'ensemble des variables l'une des valeurs de D_i jusqu'à ce que cela ne soit plus possible (voir figure 3.9).

La décomposition par les valeurs a été adopté par la majorité des approches [16, 41, 1, 22] lesquelles affectent les nuits dans une première phase.

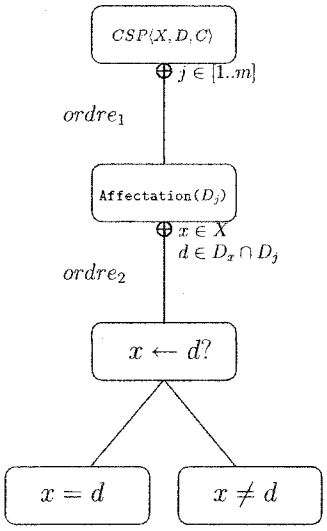


Figure 3.8 – Décomposition par les variables.

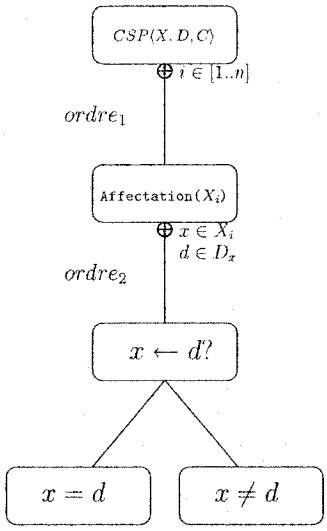


Figure 3.9 – Décomposition par les valeurs.

CHAPITRE 4

MODÈLE DE PROGRAMMATION PAR CONTRAINTES

Nous proposons dans ce chapitre un modèle CSP du problème décrit au chapitre deux. Nous discutons donc ici uniquement des règles dures. La modélisation des règles souples sera abordé dans le chapitre cinq.

La modélisation des règles dures nécessite un peu d'attention. En PC, les contraintes sont construites à partir d'un langage de base contenant les contraintes atomiques unaires et binaires usuelles ainsi que des contraintes disposant d'algorithmes de filtrage spécifiques. Il est possible qu'une règle puisse s'exprimer de plusieurs façons, en combinant les contraintes du langage de base différemment. Si ces expressions ont la même sémantique, elles n'ont pas pour autant les mêmes caractéristiques de filtrage. Il n'existe pas de résultats théoriques indiquant la meilleure façon de modéliser un problème en CSP. Généralement il faut essayer de trouver un équilibre entre la qualité et la complexité du filtrage. Quelques règles empiriques sont néanmoins toujours suivies, comme l'utilisation systématique des contraintes globales qui fournissent des algorithmes de filtrage à la fois performants et peu coûteux.

Dans ce chapitre nous utilisons des lettres majuscules dans une police de caractères de type machine à écrire (par exemple $\mathtt{X}$) pour désigner les variables du modèle. Lorsqu'il s'agit de vecteurs de variables, nous utilisons la même police mais en gras (par exemple $\mathbf{X}$). De plus, pour simplifier les expressions mathématiques, nous utilisons la

notation $D_{\mathbf{X}}$ pour désigner le produit cartésien des domaines des variables du vecteur $\mathbf{X}$ $(D_{\mathtt{X}_1} \times \ldots \times D_{\mathtt{X}_n})$.

4.1 Langage de modélisation

Pour modéliser notre problème, nous utilisons un sous-ensemble du langage du système de contraintes sur les variables de domaine fini. Notre langage contient toutes les contraintes atomiques, unaires ou binaires, usuelles ($<$, $\leq$, $=$, $\neq$, $\in$, etc.) ainsi que les contraintes n-aires connues dans la littérature : SUM,DISTRIBUTE, STRETCH, PATTERN, COND_PATTERN et EXT. Pour les contraintes atomiques, l'algorithme de filtrage maintient la cohérence de noeud ou la cohérence d'arc, suivant que la contrainte est unaire ou binaire. Dans les sections suivantes, nous décrivons la sémantique des contraintes $n-$aires de notre langage en précisant à chaque fois la complexité de leur algorithme de filtrage.

4.1.1 La contrainte SUM

Soient un vecteur $\mathbf{X} = (\mathtt{X}_1, \mathtt{X}_2, \ldots, \mathtt{X}_n)$ de variables de domaine fini et une variable Y supplémentaire prenant toutes leurs valeurs dans l'ensemble des entiers relatifs. La contrainte $\mathtt{SUM}\langle\mathbf{X}, \mathtt{Y}\rangle$ assure que Y est la somme des variables $\mathtt{X}_i$, c'est-à-dire :

$$T_{\mathtt{SUM}\langle\mathbf{X}, \mathtt{Y}\rangle} = \left\{ (s,t) \in D_{\mathbf{X}} \times D_{\mathtt{Y}} \;\middle|\; t = \sum_{i=1}^{n} s_i \right\}.$$

L'algorithme de filtrage de cette contrainte maintient seulement la cohérence de borne mais a l'avantage d'avoir une complexité linéaire.

4.1.2 La contrainte DISTRIBUTE

Soient un vecteur $\mathbf{X} = (\mathtt{X}_1, \mathtt{X}_2, \ldots, \mathtt{X}_n)$ de variables de domaine fini prenant leurs valeurs dans l'ensemble $V = \{v_1, v_2, \ldots, v_m\}$ et un vecteur $\mathbf{C} = (\mathtt{C}_1, \mathtt{C}_2, \ldots, \mathtt{C}_m)$ de variables *de comptage*, de domaine fini également, prenant leurs valeurs dans l'ensemble des entiers naturels. La contrainte $\mathtt{DISTRIBUTE}(\mathtt{C}, V, \mathtt{X})$ assure que chaque $\mathtt{C}_j$ est égale au nombre de fois que la valeur v_j est affectée à une variable de X, c'est-à-

dire :

$$T_{\text{DISTRIBUTE}(\mathtt{C},\, V,\, \mathtt{X})} = \left\{ (s,t) \in D_{\mathtt{C}} \times D_{\mathtt{X}} \;\middle|\; \forall j \in [1..m],\ s_j = \left| \{ i \in [1..n] \mid t_i = v_j \} \right| \right\}.$$

Sémantiquement, cette contrainte est équivalente à m contraintes de cardinalité simple. Cependant, leur traitement simultané permet l'utilisation d'un algorithme de filtrage polynomial [40] capable de maintenir la cohérence d'arc généralisée. Il est donc important, lors de la modélisation, de regrouper au maximum les contraintes de cardinalité pour tirer profit de cette contrainte globale.

Remarque 2 *: Lorsque $V = [1..m]$, on écrit seulement* DISTRIBUTE(**C**, **X**).

4.1.3 La contrainte STRETCH

Définition 4.1 (Séquence) *Soit $(s_1,\ s_2,\ \ldots,\ s_n)$ un vecteur d'éléments de V. On appelle* séquence (maximale) de type $v \in V$ *toute suite $s_i,\ s_{i+1}, \ldots,\ s_j$ d'éléments consécutifs telle que :*

$$\begin{cases} s_i = s_{i+1} = \ldots = s_j = v \\ i > 1 \Rightarrow s_{i-1} \neq v \\ j < n \Rightarrow s_{j+1} \neq v. \end{cases}$$

La taille d'une telle séquence est alors $j - i + 1$.

Soit un vecteur $\mathtt{X} = (\mathtt{X}_1,\ \mathtt{X}_2,\ \ldots,\ \mathtt{X}_n)$ de variables de domaine fini prenant leurs valeurs dans l'ensemble $V = \{v_1,\ v_2,\ \ldots,\ v_m\}$. Soient $\underline{\lambda}$ et $\overline{\lambda}$ deux vecteurs d'entiers de taille m. La contrainte STRETCH(**X**, V, $\underline{\lambda}$, $\overline{\lambda}$) assure que dans toute affectation valide de **X** la taille des séquences de type v_k, $1 \leq k \leq m$, est comprise entre $\underline{\lambda}_k$ et $\overline{\lambda}_k$. Cette contrainte dispose d'un algorithme de filtrage de complexité polynomiale [36].

Remarque 3 *: Lorsque $V = [1..m]$, on écrit seulement* STRETCH(**X**, $\underline{\lambda}$, $\overline{\lambda}$).

4.1.4 La contrainte PATTERN

Définition 4.2 ($k-$patron) *On appelle* k-patron *toute suite de k éléments appartenant à un ensemble V quelconque telle que deux éléments consécutifs sont différents.*

Soit $\mathbf{s} = (s_1, s_2, \ldots, s_n)$ un vecteur d'éléments de $V = \{v_1, v_2, \ldots, v_m\}$. Considérons la suite $v_{i_1}, v_{i_2}, \ldots, v_{i_\ell}$ des types des séquences apparaissant successivement dans ce vecteur. Notons que, d'après la définition d'une *séquence*, deux types consécutifs sont forcément différents. Soit $\mathcal{P}_k$ un ensemble de $k-$patrons. On dit que le vecteur $\mathbf{s}$ est $\mathcal{P}_k-$valide si et seulement si toute sous-suite de k types (des séquences du vecteur $\mathbf{s}$) consécutifs correspond à un patron de $\mathcal{P}_k$.

Exemple 19 *: Considérons le vecteur* $\mathbf{a} = (a, a, b, b, b, a, c, c)$. *La suite des types de séquence correspondant est* a, b, a, c. *Pour que* $\mathbf{a}$ *soit* $\mathcal{P}_3-$*valide, il faut que* $\mathcal{P}_3$ *contienne au moins les* $3-$*patrons apparaissant dans la suite* a, b, a, c, *c'est-à-dire : aba et bac.*

Soient un vecteur $\mathbf{X} = (\mathtt{X}_1, \mathtt{X}_2, \ldots, \mathtt{X}_n)$ de variables de domaine fini, prenant leurs valeurs dans un ensemble V, et un ensemble de $k-$patrons $\mathcal{P}_k$. La contrainte PATTERN($\mathbf{X}$, $\mathcal{P}_k$) signifie que toute assignation de $\mathbf{X}$ doit être $\mathcal{P}_k-$valide. Il existe également un algorithme de filtrage polynomial [37] pour cette contrainte.

4.1.5 La contrainte COND_PATTERN

Soit $\mathbf{X} = (\mathtt{X}_1, \mathtt{X}_2, \ldots, \mathtt{X}_n)$ un vecteur de variables de domaine fini, prenant leurs valeurs dans un ensemble V. La contrainte COND_PATTERN(X, n_1, v_1, n_2, v_2) signifie que dans toute assignation de X toute séquence de type $v_1 \in V$ doit, d'une part, être de taille inférieure à n_1 et, d'autre part lorsque celle-ci est de taille n_1, être immédiatement suivie d'une séquence de type $v_2 \in V$ de taille au moins égale à n_2. Cette contrainte peut s'exprimer à l'aide de la contrainte STRETCH et de la contrainte PATTERN (voir [36]). Son algorithme de filtrage est donc polynômial.

Exemple 20 *: Considérons la contrainte* COND_PATTERN(X, 3, a, 2, b). $(b, b, c, a, a, a, a, b, b)$ *n'est pas une assignation valide car elle contient une séquence de type a de taille 4. L'assignation* $(a, a, c, c, a, a, a, b, a)$ *n'est pas valide non plus car la séquence de type a de taille 3 est suivie d'une séquence de type b trop courte. L'assignation* $(a, c, c, a, a, a, b, b, b)$ *est quant à elle valide.*

4.1.6 La contrainte EXT

La contrainte EXT($\mathbf{X}$, $\mathcal{T}$) définit l'ensemble $\mathcal{T} \subset \mathcal{P}(D^{\circ}_{\mathbf{X}})$ des tuples admissibles. Elle peut donc exprimer n'importe quelle relation entre les variables $\mathbf{X}$. La particularité de cette contrainte provient de son algorithme de filtrage associé qui maintient la cohérence d'arc généralisée (définition 3.3). Comme la complexité de cet algorithme est exponentielle, on utilise la contrainte EXT uniquement lorsque le nombre de variables est réduit.

4.2 Modélisation du problème

Les jours de l'horizon de planification sont indexés par les éléments de l'ensemble $H = \{1, 2, \ldots, h\}$. On appelle période de l'horizon tout sous-ensemble I de H. On définit aussi $W = \{1, 2, \ldots, w\}$ l'ensemble des week-ends de l'horizon. Le personnel est représenté par l'ensemble $P = \{1, 2, \ldots, p\}$. Les quarts sont représentés par l'ensemble $Q = \{1, 2, \ldots, q\}$. Cet ensemble contient deux quarts fictifs qui permettent de simplifier la modélisation. Il s'agit du quart `off` $(q-1)$ caractérisant une journée de congé et du quart `vac` (q) caractérisant une journée de vacances. On parlera de type de quart pour désigner tout sous-ensemble de Q. Par exemple, le type `travail` désigne l'ensemble $Q \setminus \{\texttt{off}, \texttt{vac}\}$, le type `non-travail`, l'ensemble $\{\texttt{off}, \texttt{vac}\}$. On peut ainsi définir les types `jour`, `soir`, `nuit`, ainsi que tout autre type spécifique à un contexte.

4.2.1 Les variables

Nous utilisons trois types de variables. Les variables de décision du problème que l'on nomme *variables d'affectation* ainsi que des *variables de week-end* et des *variables de type* qui permettent de rendre plus concise et plus simple la modélisation de certaines règles.

4.2.1.1 Les variables d'affectation

Pour chaque individu $i \in P$ et pour chaque jour $j \in H$ de l'horizon, on définit une variable $\mathtt{X}_{ij}$ à valeur dans Q indiquant le quart effectué par cette personne durant le jour j. Souvent on aura besoin de considérer seulement les variables d'un horaire

individuel i, que l'on notera $\mathbf{X}_{i\bullet} = (\mathtt{X}_{i1}, \mathtt{X}_{i2}, \ldots, \mathtt{X}_{ih})$, ou sur les variables d'un jour j donné, que l'on notera $\mathbf{X}_{\bullet j} = (\mathtt{X}_{1j}, \mathtt{X}_{2j}, \ldots, \mathtt{X}_{pj})$.

4.2.1.2 Les variables de week-end

Pour modéliser certaines règles concernant les week-ends, nous introduisons une variable supplémentaire $\mathtt{W}_{ik}$ pour chaque membre du personnel i et chaque week-end k de l'horizon. Ces variables sont à valeur dans $\{\top, \bot\}$ et indiquent si le week-end k est travaillé ($\top$) ou non ($\bot$) par la personne i. On dit qu'un week-end est travaillé dès que l'un des quarts du week-end est travaillé. Soient $J^k \subset H$ l'ensemble des jours de l'horizon correspondant au week-end k et Q_j^k, $j \in J^k$, les ensembles de quarts, des jour j du week-end, comptant pour des quarts de week-end. La contrainte qui lie les variables de week-end aux variables de décision s'écrit :

$$\mathtt{W}_{ik} = \top \quad \Leftrightarrow \quad \bigvee_{j \in J^k} \bigvee_{v \in Q_j^k} (\mathtt{X}_{ij} = v)\,.$$

On peut aussi représenter ces contraintes en extension puisqu'elles ne portent que sur un nombre limité de variables. Si l'on note $\mathcal{T}_\bot = \bigotimes_{j \in J^k} Q_j^k$ l'ensemble des tuples des variables $(\mathtt{X}_{ij})_{j \in J^k}$ du week-end k correspondant à un week-end non travaillé et $\mathcal{T}_\top = Q^{|J^k|} \setminus \mathcal{T}_\bot$, l'ensemble des tuples correspondant à des week-ends travaillés, alors l'ensemble des tuples admissibles de la contrainte est :

$$\mathcal{T} = \{\top\} \times \mathcal{T}_\top \cup \{\bot\} \times \mathcal{T}_\bot.$$

Souvent on aura besoin de travailler seulement sur les variables de week-end d'un horaire individuel i, que l'on notera $\mathbf{W}_{i\bullet} = (\mathtt{W}_{i1}, \mathtt{W}_{i2}, \ldots, \mathtt{W}_{iw})$

4.2.1.3 Les variables de type

Pour exprimer les règles portant sur des types de quart, nous introduisons, quand cela est nécessaire, une variable auxiliaire $\mathtt{Y}$ qui indique le type d'une affectation $\mathtt{X}$ relativement à un ensemble de m types disjoints $\{T_1, T_2, \ldots, T_m\}$. $\mathtt{X}$ et $\mathtt{Y}$ sont alors reliées par la contrainte $\mathtt{TYPE}_{\langle T_1, \ldots, T_m \rangle}(\mathtt{X}, \mathtt{Y})$ dont la sémantique est :

$$T_{\mathtt{TYPE}_{\langle T_1, \ldots, T_m \rangle}(\mathtt{X},\mathtt{Y})} = \left\{(s,t) \in D_\mathtt{X} \times [0..m] \;\middle|\; s \in T_t)\right\},$$

où $T_0 = Q \setminus \cup_{1 \leq k \leq m} T_k$.

Remarque 4 *: Dans le cas où* $\mathbf{X}$ *et* $\mathbf{Y}$ *sont deux vecteurs de taille n, on a l'équivalence suivante :* $\texttt{TYPE}_{\langle T_1, \ldots, T_m \rangle}(\mathbf{X}, \mathbf{Y}) \quad \equiv \quad \forall i \in [1..n],\ \texttt{TYPE}_{\langle T_1, \ldots, T_m \rangle}(X_i, Y_i)$.

4.2.2 Les contraintes

Dans cette section, chaque règle dure du problème est exprimée à l'aide du langage défini à la section 4.1. Ces expressions définissent à la fois une sémantique formelle et un algorithme de filtrage pour chaque contrainte. Nous expliquons autant que possible les choix de modélisation effectués en termes de qualité et complexité du filtrage.

4.2.2.1 Les contraintes de demande [DEM]

Dans le cas général, la demande est exprimée sur des ensembles de quarts non forcément disjoints (voir section 2.5.1). On ne peut donc pas utiliser la contrainte globale DISTRIBUTE sur ces sous-ensembles. Il faut cependant s'assurer que dans le cas particulier des médecins, dans lequel la demande s'exprime sur chacun des différents quarts, la contrainte DISTRIBUTE est bien utilisée.

Soient $j \in H$ un jour de l'horizon, $\{T_1,\ T_2,\ \ldots,\ T_m\}$ les ensembles de quarts caractérisant les périodes de quotas du jour j (voir section 2.5.1), $I \subset P$ un sous-ensemble du personnel, ainsi que deux vecteurs de taille m, $\underline{q}$ et $\overline{q}$, indiquant les quotas minima et maxima pour chaque période. Les variables intermédiaires $\mathbf{M} = (\mathtt{M}_1,\ \ldots,\ \mathtt{M}_q)$ indiquent le nombre de fois que chaque quart est planifié au cours de la journée j pour un individu du sous-ensemble I. Les variables $\mathtt{S} = (\mathtt{S}_1,\ \ldots,\ \mathtt{S}_m)$ indiquent, pour chaque période de quota, l'effectif du personnel de I qui travaille. Voici comment on exprime la demande du jour j pour le sous-ensemble du personnel I :

$$
\begin{array}{l}
\texttt{DISTRIBUTE}(\mathbf{M},\ Q,\ \mathbf{X}_{\bullet j|_I}) \quad \wedge \\
\forall \ell \in [1..m], \\
\quad \left| \begin{array}{l} \texttt{SUM}(\mathbf{M}_{|_{T_\ell}},\ \mathtt{S}_\ell) \quad \wedge \\ \underline{q}_\ell \leq \mathtt{S}_\ell \leq \overline{q}_\ell. \end{array} \right.
\end{array}
$$

4.2.2.2 Pré-affectations et affectations interdites [AVA1]

Soient $A_I \subset \mathcal{P}(P \times H \times Q)$ l'ensemble des affectations interdites et $A_O \subset \mathcal{P}(P \times H \times Q)$ l'ensemble des affectations obligatoires. Ces contraintes unaires constituent

un pré-traitement :

$$\forall (i,j,q) \in A_I, \quad \mathtt{X}_{ij} \neq q \quad \wedge \quad \forall (i,j,q) \in A_O, \quad \mathtt{X}_{ij} = q.$$

4.2.2.3 Contraintes sur la charge horaire [WOR]

Les quarts de travail sont souvent de durée variable (quatre heures, huit heures, dix heures, etc.). Même si le nombre de durées différentes varie selon le contexte, il n'excède jamais cinq. En partant de ce constat, nous avons traité les contraintes en passant par les multiplicités des quarts de même durée car il est possible ensuite d'utiliser la cohérence d'arc généralisée sur l'ensemble des variables de multiplicité (variables comptant le nombre de quart de chaque durée).

Soient $i \in P$ un membre du personnel, J une période de l'horizon, une partition $\langle T_1, \ldots, T_\ell \rangle$ de l'ensemble des quarts en types regroupant les quarts de même durée, $\mathtt{M} = (\mathtt{M}_1, \ldots, \mathtt{M}_\ell)$ les variables de multiplicité correspondantes et $\mathcal{T}$ l'ensemble des tuples admissibles pour les variables $\mathtt{M}$. Si les $(h_k)_{1 \leq k \leq \ell}$ désignent les durées des quarts de type T_k et $\underline{h}$ et $\overline{h}$ indiquent la charge horaire minimum et maximum de l'individu i sur la période J, alors l'ensemble des tuples admissibles se calcule de la manière suivante :

$$\mathcal{T} = \left\{ m \in [0..\overline{m}_1] \times \ldots \times [0..\overline{m}_\ell] \;\middle|\; \underline{h} \leq \sum_{1 \leq k \leq \ell} h_k m_k \leq \overline{h} \right\},$$

où $\overline{m}_k = \lfloor \overline{h}/h_k \rfloor$. La contrainte s'écrit alors :

$$\begin{aligned} &\mathtt{TYPE}_{\langle T_1, \ldots, T_\ell \rangle}(\mathtt{X}_{i\bullet|_J}, \mathtt{Y}) \quad \wedge \\ &\mathtt{DISTRIBUTE}(\mathtt{M}, \mathtt{Y}) \quad \wedge \\ &\mathtt{EXT}(\mathtt{M}, \mathcal{T}). \end{aligned}$$

Exemple 21 *: Prenons le contexte des unités de soins infirmiers où les quarts durent 4, 8 ou 12 heures et la charge de travail est de 80 heures par période de deux semaines. Pour un individu i, on va donc avoir trois variables de multiplicité $\mathtt{M}_1$, $\mathtt{M}_2$ et $\mathtt{M}_3$, comptant respectivement le nombre de quarts de 4 heures, de 8 heures et de 12 heures dans la partie de l'horaire de i correspondant à la période de deux semaines. L'ensemble des tuples admissibles est égal à l'ensemble des solutions positives de l'équations diophantienne $4m_1 + 8m_2 + 12m_3 = 80$.*

4.2.2.4 Distribution d'un type de quart sur un période [DIS2]/[FAI]

Cette contrainte permet d'exprimer plusieurs règles qu'il est important de regrouper, une fois encore, afin d'utiliser au mieux la contrainte globale DISTRIBUTE. Nous allons regrouper toutes les contraintes portant sur la même période de l'horizon et sur des types appartenant à une même partition de l'ensemble des quarts (jour/soir/nuit, travail/congé/vacances, cube/choc/suite de soins).

Soient $i \in P$ un membre du personnel, J une période de l'horizon, $\langle T_1, \ldots, T_\ell \rangle$ une partition de Q, $\mathtt{M} = (\mathtt{M}_1, \ldots, \mathtt{M}_\ell)$ les variables comptant le nombre de quarts de chaque type au cours de la période J, $\underline{m}$ et $\overline{m}$ deux vecteurs de taille ℓ indiquant les nombres minima et maxima de quarts de chaque type sur la période J pour l'individu i. Les contraintes s'écrivent :

$$\begin{aligned} &\mathtt{TYPE}_{\langle T_1, \ldots, T_\ell \rangle}(\mathbf{X}_{i\bullet|_J}, \mathbf{Y}) \quad \wedge \\ &\mathtt{DISTRIBUTE}(\mathbf{M}, \mathbf{Y}) \quad \wedge \\ &\forall k \in [1..\ell], \quad \underline{m}_k \leq \mathtt{M}_k \leq \overline{m}_k. \end{aligned}$$

4.2.2.5 Distribution des week-ends [ERG4]

Soient $i \in P$ un membre du personnel et $\underline{n}$ et $\overline{n}$ les nombres minima et maximums de week-ends travaillés et non travaillés consécutifs autorisés par i. La contrainte s'exprime simplement avec la contrainte globale STRETCH :

$$\mathtt{STRETCH}(\mathbf{W}_{i\bullet}, \{\top, \bot\}, \underline{n}, \overline{n}).$$

4.2.2.6 Patrons de jours sur une période donnée [ERG1]

Soient $i \in P$ un membre du personnel, $J \subset H$ une période de l'horizon (typiquement un week-end) et $\mathcal{P}_J$ l'ensemble des tuples admissibles pour les variables d'affectation de l'individu i sur la période J i.e. l'ensemble des patrons autorisés. La contrainte s'écrit :

$$\mathtt{EXT}(\mathbf{X}_{i\bullet|_J}, \mathcal{P}_J).$$

4.2.2.7 Patrons de séquences [ERG2]

Soient $i \in P$ un membre du personnel et $\mathcal{P}$ un ensemble de patrons de séquences d'arités quelconques. Cette contrainte est l'application directe de la contrainte globale PATTERN :

$$\texttt{PATTERN}(\mathbf{X}_{i\bullet},\ \mathcal{P}).$$

4.2.2.8 Affectations consécutives [ERG3]

Soient $i \in P$ un membre du personnel, $\{T_1,\ T_2,\ \ldots,\ T_\ell\}$ un ensemble de types de quart disjoints et $\underline{n}$ et $\overline{n}$ deux vecteurs de taille ℓ indiquant les tailles minimums et maximums des séquences de quarts de chaque type dans l'horaire de l'individu i. La contrainte s'exprime simplement à l'aide de la contrainte globale STRETCH :

$$\begin{aligned}&\texttt{TYPE}_{\langle T_1,\ \ldots,\ T_\ell\rangle}(\mathbf{X}_{i\bullet},\ \mathbf{Y}) \quad \wedge\\ &\texttt{STRETCH}(\mathbf{Y},\ \underline{n},\ \overline{n}).\end{aligned}$$

4.2.2.9 Séquences consécutives [ERG5]

Cette contrainte n'exprime qu'une partie des règles **[ERG5]**. Celles du type : *«accorder au moins n_2 quarts consécutifs de type T_2 après une séquence de plus de n_1 quarts de type T_1»*. Elle s'exprime simplement, pour l'individu i, à l'aide de la contrainte globale COND_PATTERN :

$$\begin{aligned}&\texttt{TYPE}_{\langle T_1,\ T_2,\ Q\setminus(T_1\cup T_2)\rangle}(\mathbf{X}_{i\bullet},\ \mathbf{Y}) \quad \wedge\\ &\texttt{COND\_PATTERN}(\mathbf{Y},\ n_1,\ 1,\ n_2,\ 2).\end{aligned}$$

4.2.2.10 Distribution entre plusieurs types de quarts [ERG7]

Soient n types de quarts disjoints $(T_k)_{1\leq k\leq n}$ ainsi que n intervalles $([\underline{r}_k, \overline{r}_k])_{1\leq k\leq n}$ inclus dans [0,1]. La contrainte impose que le rapport *quarts de type T_k/quarts travaillés* sur la période soit compris entre $\underline{r}_k$ et $\overline{r}_k$.

$$\begin{aligned}&\texttt{TYPE}_{\langle T_1,\ \ldots,\ T_n\rangle}(\mathbf{Y},\ \mathbf{X}_{i\bullet}) \quad \wedge\\ &\texttt{DISTRIBUTE}(\mathbf{M},\ \mathbf{Y}|_J) \quad \wedge\\ &\texttt{SUM}(\mathbf{M},\ \mathtt{N}) \quad \wedge\\ &\forall k \in [1..n], \quad \underline{r}_k\mathtt{N} \leq \mathtt{M}_k \leq \overline{r}_k\mathtt{N}.\end{aligned}$$

Remarque 5 *: pour l'instant, cette contrainte n'est implantée que pour la partition de type* `jour/soir/nuit`.

CHAPITRE 5

STRATÉGIES ET HEURISTIQUES DE RECHERCHE

On appelle stratégie de recherche l'ensemble des mécanismes mis en oeuvre afin d'utiliser en pratique l'algorithme de *backtracking* (voir algorithme 3.1) et de *branch & bound* (voir algorithme 3.2). Comme nous le mentionnons dans la section 3.2.4, il n'existe pas de résultats théoriques sur l'élaboration des stratégies de recherche. Notre démarche ne s'appuie donc que sur des travaux existants [16, 11, 31, 1, 22, 41, 19] souvent très brefs dans la description de leurs heuristiques, ainsi que sur divers résultats empiriques. Le formalisme utilisé dans ce chapitre est expliqué à la section 3.2.4.1.

5.1 Modèle général

Nous présentons ici un cadre formel permettant de définir des heuristiques de choix de couples variable/valeur dont la caractéristique majeure est d'avoir une structure modulaire calquée sur celle du modèle (voir figure 5.1). À chaque contrainte du modèle (dure ou souple), on peut associer une heuristique appelée *heuristique d'incitation*, dont le but est de favoriser parmi les variables futures (i.e. les variables non encore instanciées), des affectations qui aident à satisfaire la contrainte si celle-ci est dure, ou qui mènent à de bonnes solutions relativement au critère défini par la contrainte,

si celle-ci est souple. Les heuristiques d'incitation favorisent ou défavorisent une affectation potentielle en lui associant un poids.

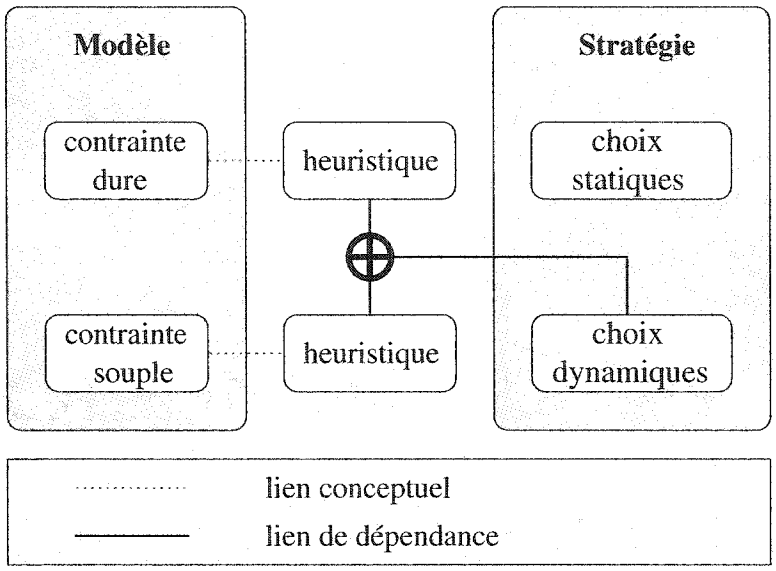


Figure 5.1 – Structure de la stratégie

Définition 5.1 (affectation potentielle) *Une affectation potentielle est un couple variable/valeur* (X, v) *tel que* $v \in D_X$. *On note* $A_X = \{X\} \times D_X$ *l'ensemble des affectations potentielles de la variable* X. *Si* V *est un ensemble de variables, alors* A_V *désigne l'ensemble des affectations potentielles de toutes les variables de* V.

5.1.1 Choix statiques

Les choix statiques sont établis en fonction de la décomposition du problème. Nous discutons des divers choix statiques lorsque nous présentons nos stratégies concrètes à la section 5.2.

5.1.2 Choix dynamiques

Ce sont les heuristiques d'incitations qui déterminent les choix dynamiques en mettant à jour après chaque phase de propagation les poids pour les affectations potentielles. Elles sont également à la base de la modularité des stratégies. À chaque

contrainte du modèle, l'utilisateur peut associer une heuristique d'incitation et décider de l'activer ou non.

Définition 5.2 (heuristique d'incitation) *Une heuristique d'incitation h dépend d'un sous-ensemble V_h des variables du problème. Elle favorise ou défavorise certaines affectations potentielles en fonction de l'état courant de ces variables (défini par les domaines courants). Pour un état donné des variables de V_h, on note $\phi_h \colon A_{V_h} \xmapsto{\phi_h} \mathbb{R}$ la fonction qui associe les incitations aux affectations.*

Soient $\mathcal{H}$ l'ensemble des heuristiques d'incitation prises en compte et X l'ensemble des variables de décision $\mathtt{X}_{ij}$ du problème. Pour toute affectation a de A_X, on peut calculer un poids :

$$\pi(a) = \sum_{h \in \mathcal{H}} \omega_h \phi_h(a),$$

où $(\omega_h)_{h \in \mathcal{H}}$ est une pondération des heuristiques. L'affectation choisie par les choix dynamiques est alors celle de poids maximal.

Il se peut néanmoins qu'une heuristique incite des affectations de variables auxiliaires du problème. Pour intégrer ces incitations avec les autres, nous allons devoir les convertir en incitations sur des affectations de variables de décision. Nous le faisons par l'intermédiaire d'une fonction $\tau \colon A_V \xmapsto{\tau} \mathcal{P}(A_X)$ qui fait correspondre à chaque affectation potentielle d'une variable quelconque un ensemble d'affectations potentielles des variables de décision. Cette fonction est définie à l'aide des contraintes de liaison existant entre les variables auxiliaires et les variables de décision. Pour une variable de décision $\mathtt{X}_{ij}$, τ se réduit à l'identité, i.e. :

$$\forall a \in A_{\mathtt{X}_{ij}}, \quad \tau(a) = \{a\}.$$

Pour une variable de week-end $\mathtt{W}_{ik}$:

$$\begin{aligned} \tau(\mathtt{W}_{ik}, \top) &= \textstyle\bigcup_{j \in J^k} \bigcup_{q \in Q_j^k} \{(\mathtt{X}_{ij}, q)\} \quad \text{et} \\ \tau(\mathtt{W}_{ik}, \bot) &= \textstyle\bigcup_{j \in J^k} \bigcup_{q \in Q \setminus Q_j^k} \{(\mathtt{X}_{ij}, q)\}. \end{aligned}$$

Pour une variable de type $\mathtt{Y}$ relative à la partition de types $\langle T_1, T_2, \ldots, T_m \rangle$:

$$\forall k \in [1..m], \quad \tau(\mathtt{Y}, k) = \bigcup_{q \in T_k} \{(\mathtt{X}_{ij}, q)\}.$$

Finalement, le calcul des poids est un peu plus compliqué dans le cas général :

$$\pi(a) \quad = \quad \sum_{h \in \mathcal{H}} \omega_h \left(\sum_{a' \in A_{V_h} \text{t.q. } a \in \tau(a')} \phi_h(a') \right).$$

Remarque 6 *: Les poids ω_h servent à hierarchiser les heursistiques et sont fixés par l'utilisateur.*

5.1.3 Quelques heuristiques d'incitation

Dans les sections suivantes, nous définissons des heuristiques d'incitation pour les contraintes de cardinalité (respect de la demande, contraintes d'équité, respect de la charge de travail) et pour les contraintes somme (respect de la charge de travail).

5.1.3.1 Heuristique pour la contrainte de cardinalité

Une contrainte de cardinalité est un quadruplet $C = (Y,\ t,\ \underline{\nu},\ \overline{\nu})$ où $Y = \{\mathtt{Y}_1,\ \mathtt{Y}_2,\ \ldots,\ \mathtt{Y}_m\}$ est un ensemble de variables de domaines finis prenant leur valeur dans l'ensemble T (ensemble de types), t est un élément de T (un type) et $\underline{\nu}$ et $\overline{\nu}$ sont deux entiers naturels. Elle impose que le nombre de variables prenant la valeur t soit compris entre $\underline{\nu}$ et $\overline{\nu}$.

Considérons une instanciation partielle des variables de X vérifiant la cohérence de borne pour une contrainte de cardinalité. Nous allons tenter de définir intuitivement une quantité $\tilde{n}$ qui estime le nombre de variables égales à t vers lequel le système tend. Pour cela, considérons les quantités $\underline{n}$ et $\overline{n}$ qui comptent respectivement le nombre de variables de Y dont le domaine courant est $\{t\}$ (variables qui comptent en positif) et le nombre de variables dont le domaine courant ne contient pas t (variables qui comptent en négatif). Ainsi on a :

$$\begin{aligned}
\underline{n} &= |\{\mathtt{X} \in X \mid D_{\mathtt{X}} = \{t\}\}| \\
\overline{n} &= m - |\{\mathtt{X} \in X \mid t \notin D_{\mathtt{X}}\}| \\
\tilde{n} &= \tfrac{\overline{n}-\underline{n}}{2}.
\end{aligned}$$

Voici les trois cas que l'on peut alors rencontrer (ils sont illustrés par la figure 5.2) :

- Le cas $\tilde{n} < \underline{\nu}$, dans lequel il faut favoriser les affectations de la valeur t. La fonction d'incitation ϕ_h se définit donc pour toute variable $\mathtt{X}$ et pour toute

valeur v par :

$$\phi_h(\mathtt{X}, v) = \begin{cases} 1 & \text{si } v = t, \\ 0 & \text{sinon.} \end{cases}$$

- Le cas $\underline{\nu} \leq \tilde{n} \leq \overline{\nu}$, dans lequel on ne favorise aucune affectation. La fonction d'incitation ϕ_h se définit donc pour toute variable $\mathtt{X}$ et pour toute valeur v par :

$$\phi_h(\mathtt{X}, v) = 0.$$

- Le cas $\tilde{n} > \overline{\nu}$, dans lequel il faut défavoriser les affectations de la valeur t. La fonction d'incitation ϕ_h se définit donc pour toute variable $\mathtt{X}$ et pour toute valeur v par :

$$\phi_h(\mathtt{X}, v) = \begin{cases} 0 & \text{si } v = t, \\ 1 & \text{sinon.} \end{cases}$$

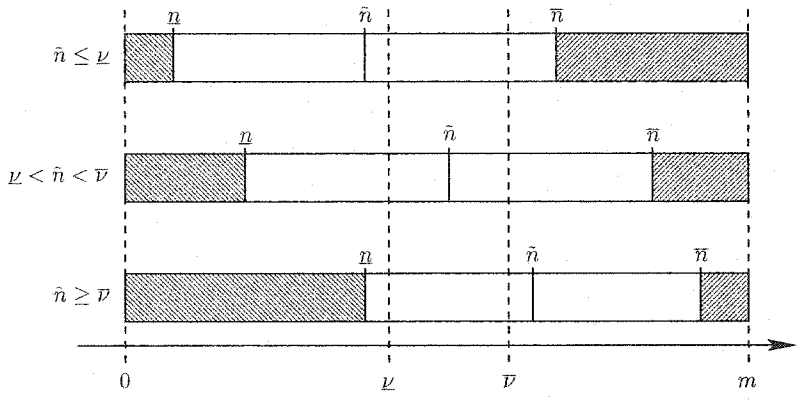


Figure 5.2 – États de l'heuristique de la contrainte de cardinalité.

5.1.3.2 Heuristique pour la contrainte somme

Une contrainte somme est définie par un triplet $(Y, \underline{\nu}, \overline{\nu})$ où $Y = \{\mathtt{Y}_1, \mathtt{Y}_2, \ldots, \mathtt{Y}_m\}$ est un ensemble de variables de domaines finis inclus dans $\mathbb{N}$ et $[\underline{\nu}, \overline{\nu}]$ un intervalle de $\mathbb{N}$. Elle impose que la somme des élément de Y soit comprise entre $\underline{\nu}$ et $\overline{\nu}$.

La contrainte de cardinalité étudiée dans la section précédente est un cas particulier de cette contrainte. Soit $C = (Y,\ t,\ \underline{\nu},\ \overline{\nu})$ une contrainte de cardinalité. Notons $\chi_{\{T\}} \colon D_Y \overset{\chi_{\{T\}}}{\longmapsto} \{0, 1\}$ la fonction caractéristique du sous-ensemble $\{t\}$ de l'union des domaine des variables de Y. Cette contrainte peut s'exprimer sous forme de la con-

trainte somme $C' = (Y,\ \underline{\nu},\ \overline{\nu})$, où Y est l'ensemble des variables $\chi_T(\mathtt{Y})$ pour toute variable $\mathtt{Y} \in Y$.

Comme précédemment, nous allons diviser l'ensemble des instanciations partielles des variables de Y en trois sous-ensembles, mais cette fois on cherche à estimer la somme finale des éléments de Y. Pour cela, nous allons introduire les quantités $\underline{s}$ et $\overline{s}$ qui comptent respectivement le nombre minimum et le nombre maximum d'heures nécessaires pour compléter l'instanciation partielle courante. Notons :

$$\begin{aligned}\underline{s} &= \sum_{\mathtt{Y}\in Y} \min(D_{\mathtt{Y}}) \\ \overline{s} &= \sum_{\mathtt{Y}\in Y} \max(D_{\mathtt{Y}}) \\ \tilde{s} &= \tfrac{\overline{s}-\underline{s}}{2}.\end{aligned}$$

Voici les trois cas que l'on peut rencontrer :

- Le cas $\tilde{s} < \underline{\nu}$, dans lequel on va favoriser les affectations des valeurs supérieures au seuil $(\max(D_Y^\circ) - \min(D_Y^\circ))/2$, qui est la moyenne des valeurs extrèmes des domaines initiaux des variables de Y. La fonction d'incitation ϕ_h se définit donc pour toute variable $\mathtt{Y} \in Y$ et pour toute valeur $v \in D_Y$ par :

$$\phi_h(\mathtt{Y}, v) = \begin{cases} 1 & \text{si } v \geq \frac{\max(D_Y^\circ)-\min(D_Y^\circ)}{2}, \\ 0 & \text{sinon.} \end{cases}$$

- Le cas $\underline{\nu} \leq \tilde{s} \leq \overline{\nu}$, dans lequel on ne favorise aucune affectation. La fonction d'incitation ϕ_h se définit donc pour toute variable $\mathtt{Y} \in Y$ et pour toute valeur $v \in D_Y$ par :

$$\phi_h(\mathtt{Y}, v) = 0.$$

- Le cas $\tilde{s} > \overline{\nu}$, dans lequel on va favoriser les affectations des valeurs inférieures au seuil $(\max(D_Y^\circ) - \min(D_Y^\circ))/2$. La fonction d'incitation ϕ_h se définit donc pour toute variable $\mathtt{Y} \in Y$ et pour toute valeur $v \in D_Y$ par :

$$\phi_h(\mathtt{Y}, v) = \begin{cases} 0 & \text{si } v \leq \frac{\max(D_Y^\circ)-\min(D_Y^\circ)}{2}, \\ 1 & \text{sinon.} \end{cases}$$

5.2 Stratégies concrètes retenues

Nous utilisons plusieurs stratégies pour donner des solutions dans les divers contextes décrits à la section du chapitre six. Car même s'ils sont tous issus du même modèle, ces problèmes présentent des caractéristiques différentes qui nous ont permis de définir deux types de problèmes distincts : ceux relatifs aux unités de soins infirmiers et ceux relatifs aux salles d'urgence de médecins. Cependant, toutes les stratégies dérivent du modèle de la section 5.1. Leurs spécificités viennent de la décomposition choisie qui dépend, elle, du type de problème à résoudre.

5.2.1 Première phase : affectation des vacances

Les premiers choix dans un arbre de recherche sont importants car il y a peu de chance qu'ils soient remis en cause étant donnée la taille des problèmes étudiés. Les vacances sont dans la plupart des cas des pré-affectations et, quand ce n'est pas le cas, leurs possibilités de placement sont limitées. Nous pouvons supposer que le fait de les placer en premier sur l'horizon n'affecte ni la réalisabilité du problème, ni la qualité des solutions trouvées. Pour l'instant, le placement des vacances se fait de façon chronologique (on commence par le premier jour candidat et on essai ensuite le jours suivants dans l'ordre chronologique). Ce choix a pour conséquence de simplifier les heuristiques intervenant dans les choix futurs qui n'auront pas à prendre en compte le quart `vac`.

La figure 5.3 décrit le premier étage de l'ensemble des stratégies que nous utilisons. Il s'agit d'une décomposition par les valeurs : le premier sous-problème étant l'affectation des vacances et le second, l'affectation du reste des quarts, congés y compris.

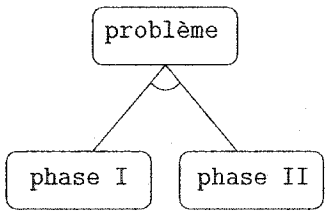


Figure 5.3 – Phases successives de la stratégie.

La stratégie de placement des vacances parcourt les membres du personnel un

par un dans un ordre $ordre_1$ indifférent et affecte ses jours de vacances dans l'ordre chronologique ($ordre_2$) jusqu'à ce que la période de vacances soit remplie ou que le nombre maximum de jours de vacances soit atteint. La figure 5.4 représente l'arbre ET/OU (section 3.2.4.1) décrivant la stratégie d'affectation des vacances.

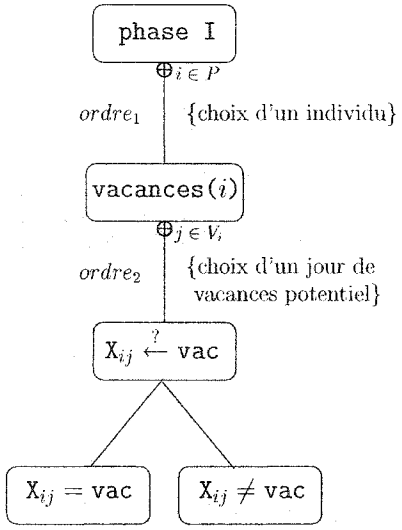


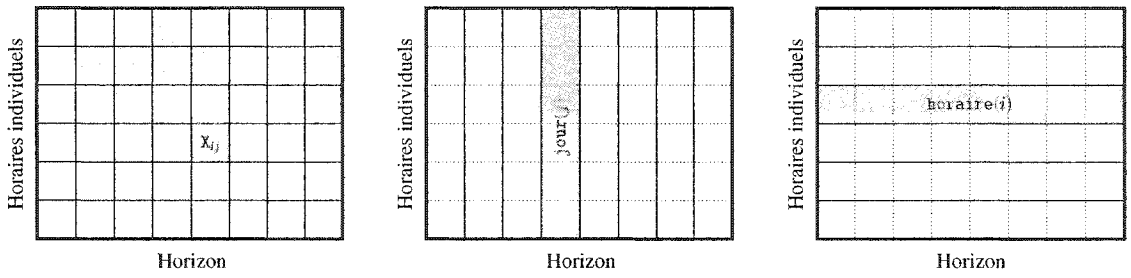
Figure 5.4 – Phase I : affectations des vacances.

5.2.2 Seconde phase

Considérons la représentation d'un horaire de travail avec en abscisse l'horizon de planification et en ordonnée les membres du personnel, chaque case correspondant à une variable de décision X_{ij} (figure 5.5(a)). On remarque alors que les variables sont liées soit verticalement, soit horizontalement. Les seules contraintes *verticales* sont les contraintes de satisfaction de la demande, le reste des contraintes étant qualifiées de *horizontales* (il s'agit des contraintes relatives aux horaires individuels). Parmi ces dernières, on distingue les contraintes de respect de la charge de travail et de distribution des quarts importuns qui s'opposent directement au critère de couverture et donc aux contraintes de respect de la demande.

Parmi l'ensemble des contextes qui ont permis l'élaboration du modèle de programmation par contraintes, on peut identifier deux familles bien distinctes de problèmes : ceux dont les contraintes verticales sont serrées (les bornes inférieures et supérieures de la demande sont égales) et donc plus contraignantes alors que la contrainte ho-

rizontale de charge de travail est lâche (les bornes inférieures et supérieures de la charge sont différentes) ; ceux dont les contraintes verticales sont lâches et les contraintes horizontales de respect de la charge de travail sont serrées. La première famille de problèmes correspond au contexte des salles d'urgence de médecins tandis que le second correspond plutôt aux unités de soins infirmiers. Pour ces deux familles de problèmes, qui sont toujours traitées séparément dans la littérature, nous proposons deux décompositions par les variables différentes.



(a) Approche classique

(b) Décomposition par jour

(c) Décomposition par horaire individuel

Figure 5.5 – Différentes décompositions du problème

La figure 5.5 illustre plusieurs façons de décomposer le problème : la façon classique où l'affectation de chaque variable constitue un sous-problème, la décomposition par jour et la décomposition par horaire individuel. La décomposition d'un problème introduit un niveau de choix supplémentaire : celui de l'ordonnancement des sous-problèmes. Dans le cas de la figure 5.5(c), il s'agit du choix d'un membre du personnel i, dans le cas de la figure 5.5(b), du choix d'un jour j de l'horizon et dans le cas de la figure 5.5(a), d'un choix de variable $\mathtt{X}_{ij}$ (jour + individu). L'introduction d'une hiérarchisation dans les choix permet d'utiliser des critères différents pour chacun des niveaux et ainsi d'affiner la stratégie.

5.2.2.1 Décomposition par horaire individuel

Cette stratégie convient bien au contexte des unités de soins infirmiers car le remplissage du planning horaire par horaire facilite la satisfaction des contraintes horizontales plus nombreuses et surtout des contraintes de respect de la charge de travail qui sont serrées dans ce cas. Cependant, les contraintes verticales lient des

variables qui seront distribuées uniformément tout au long des branches de l'arbre de recherche. Toute incohérence faisant intervenir une contrainte verticale nécessitera donc un grand nombre de retours en arrière avant d'être résolue. Il est donc nécessaire de tenir compte de ces contraintes dans les heuristiques de choix, même si le couplage est faible dans ce cas.

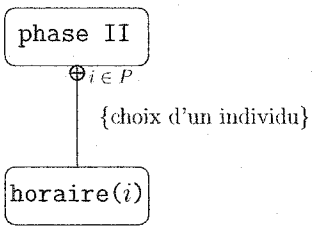
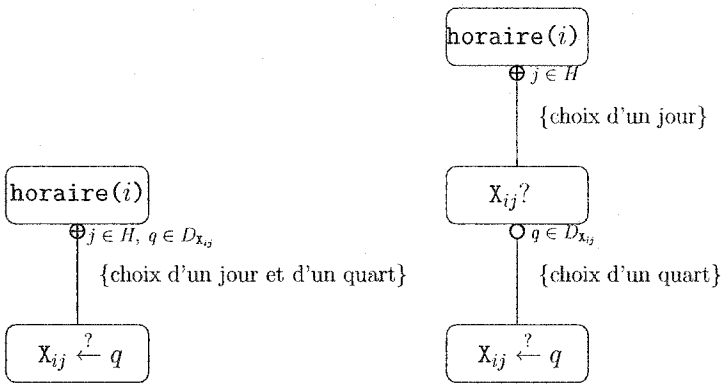


Figure 5.6 – Décomposition par horaire individuel

Cette stratégie est décrite par l'arbre ET/OU de la figure 5.6. Elle parcourt les membres du personnel dans un ordre prédéfini quelconque (aléatoire).



(a) Choix conjoint du jour et du quart

(b) Choix séparé du jour et du quart

Figure 5.7 – Stratégies pour le sous-problème `horaire(`i`)`

Pour le sous-problème `horaire(`i`)`, nous avons introduit la possibilité de choisir conjointement (figure 5.7(a)) ou séparément (figure 5.7(b)).

5.2.2.2 Décomposition par jour

Cette stratégie convient plus particulièrement au contexte de la planification des horaires de médecins en salle d'urgence. Le fait de remplir le planning jour par jour facilite la satisfaction des contraintes verticales serrées. Cependant, pour les mêmes raisons que dans la section précédente, il est nécessaire de tenir compte des contraintes horizontales dans les heuristiques de choix.

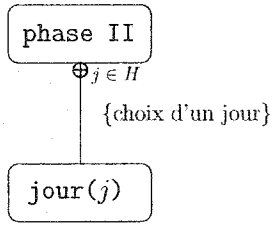
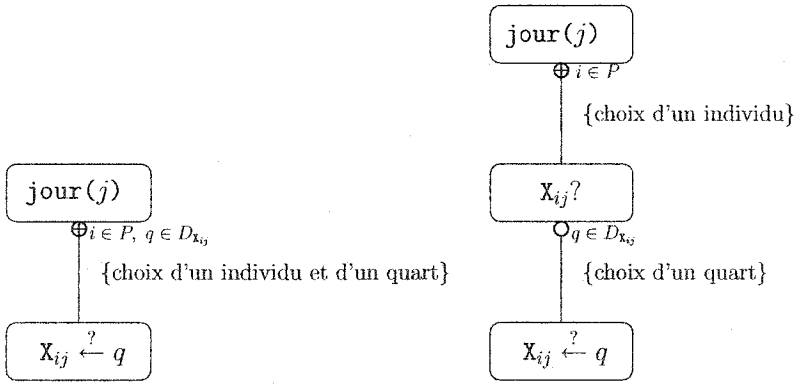


Figure 5.8 – Décomposition par jour

Cette stratégie est décrite par l'arbre ET/OU de la figure 5.8. Elle parcourt les jours de l'horizon dans l'ordre chronologique en privilégiant ou non les jours de week-end. Étant donné que l'on remplit le planning par colonnes, l'ordre chronologique nous semble être le plus pertinent pour faciliter la satisfaction des contraintes horizontales.



(a) Choix conjoint de l'individu et du quart

(b) Choix séparé de l'individu et du quart

Figure 5.9 – Stratégies pour le sous-problème `jour(`j`)`

Pour le sous-problème `jour(`j`)`, nous avons introduit la possibilité de choisir conjointement (figure 5.9(a)) ou séparément (figure 5.9(b)).

CHAPITRE 6

RÉSULTATS ET DISCUSSION

6.1 Description des données

Cette section contient des données provenant de plusieurs hôpitaux de la région de Montréal. La description de chaque contexte suit la structure du chapitre deux, avec dans un premier temps des renseignements généraux sur le personnel, les tâches et l'horizon de planification, puis une description des règles suivant la nomenclature de la section 2.4. Les règles dures seront précédées du signe ▶ et les règles souples précédées du signe ▷.

6.1.1 Unité de dialyse de l'Hôpital Royal Victoria (DIA)

Description générale

Tableau 6.1 – Description générale de l'unité de dialyse de l'Hôpital Royal Victoria.

Nom du fichier :	`DIA_sept2001.data(2)`
Effectif du personnel :	38 (dont 5 de niveau 5 et 33 de niveau 6)
Taille de l'horizon de planification :	6 semaines (42 jours)
Nombre de tâches différentes :	5

Description des tâches

Tableau 6.2 – Description des tâches de l'unité de dialyse de l'Hôpital Royal Victoria.

Symbole	Durée	Début	Fin	Période
D	8H	7 :30	15 :30	jour
E	8H	15 :30	23 :30	soir
DH	8H	11 :30	19 :30	jour
D2	12H	7 :30	19 :30	jour
E2	12H	11 :30	23 :30	soir

Description de la demande

Ici, la demande est exprimée sur des périodes. Les quarts qui couvrent chaque période sont indiqués entre parenthèses. Chaque case du tableau contient dans l'ordre les nombres de personnes minimal, voulu et maximal demandés pour la période.

Tableau 6.3 – Description de la demande de l'unité de dialyse de l'Hôpital Royal Victoria.

période	**7h30-11h30** (D,D2)	**11h30-15h30** (D,DH,D2,E2)	**15h30-19h30** (E,DH,D2,E2)	**19h30-23h30** (E,E2)
niveau 5	[0-**2**-17]	[0-**2**-17]	[0-**1**-16]	[0-**1**-9]
niveau 6	[0-**10**-18]	[0-**10**-18]	[0-**9**-19]	[2-**8**-12]

Disponibilités

- ◇ 17 semaines de vacances sont attribuées durant cet horizon, réparties comme suit : 11 infirmières ont une semaine, une a quatre semaines et une en a deux.
- ◇ Une infirmière a son horaire prédéterminé, tandis que quatre autres ont un ou deux quarts de pré-affectés par semaine.
- ◇ Les quarts D et E peuvent être travaillés par la majorité du personnel (plus de 30 infirmières), les quarts D2 et E2, par un peu plus de la moitié (entre 20 et 25) et le quart DH par deux infirmières uniquement.

Charge de travail

La charge de travail est définie par période de deux semaines et se répartit parmi le personnel comme ceci :

- ◇ 25 infirmières sont à temps complet (charge de 80 heures environ),
- ◇ le reste ont des charges allant de 24 à 64 heures.

Règles ergonomiques

- ▶ ***Week-ends non brisés*** **[ERG1]** : Cette règle s'exprime en autorisant uniquement les patrons du type `travail/travail` et `congé/congé`.
- ▶ ***Quarts du lendemain*** **[ERG2]** : L'enchaînement des quarts doit respecter la rotation `jour/soir`. Cette règle s'exprime en autorisant tous les patrons binaires du type `jour/jour` et `jour/soir`.
- ▶ ***Quarts Nombre d'affectations consécutives*** **[ERG3]** :
  - ▸ Les séquences de quarts de type `jour` sont bornées.
  - ▸ Les séquences de quarts de type `soir` sont bornées.
  - ▸ Les séquences de quarts de type `nuit` sont bornées.
  - ▸ Les séquences de jours de congé sont bornées.
  - ▸ Pas plus de sept jours de travail consécutifs.
  - ▷ Les jours de congés isolés sont à éviter : la taille minimale des séquences de jours de congés est deux.
  - ▷ Les quarts de travail isolés sont à éviter : la taille minimale des séquences de travail est deux.
- ▶ ***Week-ends consécutifs*** **[ERG4]** : Mises à part quelques exceptions, les infirmières doivent alterner un week-end travaillé sur deux.
- ▷ ***Quarts Distribution des quarts `jour/soir/nuit`*** **[ERG7]** : la plupart des infirmières veulent équilibrer leurs nombres de quart de `jour` et de `soir`.
- ▷ ***Préférences et aversions*** **[ERG8]**.

6.1.2 Unité de pédiatrie de l'Hôpital Royal Victoria (CHILD)

Description générale

Tableau 6.4 – Description générale de l'unité de pédiatrie de l'Hôpital Royal Victoria.

Nom du fichier :	`CHILD_juillet2001.data(2)`
Effectif du personnel :	41
Taille de l'horizon de planification :	6 semaines (42 jours)
Nombre de tâches différentes :	5

Tableau 6.5 – Description générale de l'unité de pédiatrie de l'Hôpital Royal Victoria.

Description des quarts de travail

Tableau 6.6 – Description des tâches de l'unité de pédiatrie de l'Hôpital Royal Victoria.

Symbole	Durée	Début	Fin	Période
D	8H	7 :15	15 :15	jour
ET	8H	17 :00	1 :00	soir
E	8H	15 :15	23 :15	soir
N	8H	23 :15	7 :15	nuit
DT	8H	10 :00	18 :00	jour

Description de la demande

Plus de détails à la section 6.1.1.

Tableau 6.7 – Description de la demande dans l'unité de pédiatrie de l'Hôpital Royal Victoria.

période	7h15-10h00 (D)	10h00-15h15 (D,DT)	15h15-17h00 (E,DT)	17h00-18h00 (DT,E,ET)	18h00-23h15 (ET,E)	23h15-1h00 (ET,N)	1h00-7h15 (N)
tous les niveaux	[5-10-22]	[5-10-22]	[1-3-6]	[2-5-8]	[1-4-7]	[2-5-8]	[1-5-10]

Disponibilités

- ◇ 19 semaines de vacances sont attribuées durant cette horizon, réparties comme suit : trois infirmières ont une semaine, cinq en ont deux semaines et deux en ont trois.
- ◇ Une infirmière a son horaire prédéterminé, tandis que six autres ont un ou deux quarts de pré-affectés par semaine.
- ◇ Les quarts D, DT et E peuvent être travaillés par la majorité du personnel (30 infirmières environs), les quarts N, par un peu plus de 10 infirmières et le quart ET par deux infirmières uniquement.

Charge de travail

La charge de travail est définie par période de deux semaines et se répartit parmi le personnel comme ceci :

- ◇ Seulement 15 infirmières sont à temps complet (charge de 80 heures environ),
- ◇ le reste ont des chargent variant de 24 à 64 heures.

Règles ergonomiques

Même chose que pour l'unité de dialyse (voir section 6.1.1).

6.1.3 Centre des naissances de l'Hôpital Royal Victoria (BC)

Description générale

Tableau 6.8 – Description générale du centre des naissances de l'Hôpital Royal Victoria.

Nom du fichier :	BC_sept2001.data(2)
Effectif du personnel :	54
Taille de l'horizon de planification :	6 semaines (42 jours)
Nombre de tâches différentes :	7

Description des quarts de travail

Tableau 6.9 – Description des tâches du centre des naissances de l'Hôpital Royal Victoria.

Symbole	Durée	Début	Fin	Période
D	8H	7 :30	15 :30	jour
DP	8H	7 :30	15 :30	jour
D12	12H	7 :30	19 :30	jour
E	8H	15 :30	23 :30	soir
S12	12H	19 :30	7 :30	nuit
N	8H	23 :30	7 :30	nuit
S	8H	23 :30	7 :30	nuit

Description de la demande

Plus de détails à la section 6.1.1.

Tableau 6.10 – Description de la demande dans le centre des naissances de l'Hôpital Royal Victoria.

période	**7h30-15h30** (D,DP,D12)	**15h30-19h30** (D12,E)	**19h30-23h30** (E,S12)	**23h30-7h30** (S12,N,S)
tous les niveaux	[5-9-21]	[5-9-21]	[5-9-21]	[5-9-21]

Disponibilités

- ◇ Cinq semaines de vacances sont attribuées durant cette horizon.
- ◇ Quatre infirmières ne travaillent que les jours de week-end et six ont une ou deux pré-affectations par semaine.
- ◇ Les quarts `D`, `E` et `N` peuvent être travaillés par la majorité du personnel (plus de 20 infirmières environs), les autres sont plus rares (moins de 10).

Charge de travail

La charge de travail est définie par période de deux semaines et se répartit parmi le personnel comme ceci :

- ◇ Seulement 17 infirmières sont à temps complet (charge de 80 heures environ),
- ◇ le reste ont des chargent variant de 24 à 64 heures.

Règles ergonomiques

Même chose que pour l'unité de dialyse (voir section 6.1.1).

6.1.4 Unité de soins de la salle d'urgence de l'Hôpital Général de Montréal (ERMGH)

Description générale

Tableau 6.11 – Description générale de l'unité de soins de la salle d'urgence de l'Hôpital Général de Montréal.

Nom du fichier :	`ERMGH_juin2002.data(2)`
Effectif du personnel :	41
Taille de l'horizon de planification :	6 semaines (42 jours)
Nombre de tâches différentes :	4

Description des quarts de travail

Tableau 6.12 – Description des tâches de l'unité de soins de la salle d'urgence de l'Hôpital Général de Montréal.

Symbole	Durée	Début	Fin	Période
D	8H	7 :30	15 :30	jour
DH	8H	12 :00	20 :00	soir
E	8H	15 :30	23 :30	soir
N	8H	23 :30	7 :30	nuit

Description de la demande

Plus de détails à la section 6.1.1.

Tableau 6.13 – Description de la demande dans l'unité de soins de la salle d'urgence de l'Hôpital Général de Montréal.

période	**7h30-12h00** (D)	**12h00-15h30** (D,DH)	**15h30-20h00** (DH,E)	**20h00-23h30** (E)	**23h30-7h30** (N)
tous les niveaux	[1-8-12]	[1-8-12]	[1-8-12]	[2-8-11]	[1-8-12]

Disponibilités

- ◇ 24 semaines de vacances sont attribuées durant cette horizon, réparties comme suit : onze infirmières ont une semaine, cinq en ont deux semaines et deux en ont trois.
- ◇ Trois infirmières ont leur horaire prédéterminé, tandis que dix autres ont entre une et dix pré-affectations sur l'horizon.
- ◇ Tous les quarts peuvent être réalisés par environ trente personnes.

Charge de travail

La charge de travail est définie par période de deux semaines et se répartit parmi le personnel comme ceci :

- ◇ Seulement 11 infirmières sont à temps complet (charge de 80 heures environ),
- ◇ le reste ont des chargent variant de 24 à 64 heures.

Règles ergonomiques

Même chose que pour l'unité de dialyse (voir section 6.1.1).

6.1.5 Hôpital Sacré-Coeur (HSCo)

Cette description s'inspire du chapitre 6 du mémoire de maîtrise de Forget [20]. S'y référer pour plus de détails.

Description générale

Tableau 6.14 – Description générale de la salle d'urgence de l'Hôpital Sacré-Coeur.

Nom du fichier :	`SacreCoeur.data(2)`
Effectif du personnel :	18 médecins dont 7 nouveaux et 11 anciens
Taille de l'horizon de planification :	1 mois (28 jours)
Nombre de tâches différentes :	9

Description des quarts de travail

Tableau 6.15 – Description des tâches de la salle d'urgence de l'Hôpital Sacré-Coeur.

Symbole	Durée	Début	Fin	Période	Type
8	8H	8 :00	16 :00	jour	choc
10	10H	8 :00	18 :00	jour	cube
16	8H	16 :00	24 :00	soir	choc
16T	8H	16 :00	24 :00	soir	cube
C[1]	8H	16 :00	24 :00	soir	cube
N	8H	0 :00	8 :00	nuit	choc
8T	8H	8 :00	16 :00	jour	cube
SS	4H	8 :00	16 :00	jour	suite de soins
18	4H	18 :00	22 :00	soir	cube

[1]L'activité de cours est donnée une fois par semaine le mercredi à la place du quart 16T.

Description de la demande

Tableau 6.16 – Description de la demande pour la salle d'urgence de l'Hôpital Sacré-Coeur.

Quarts	Lun	Mar	Mer	Jeu	Ven	Sam	Dim
8	✓	✓	✓	✓	✓	✓	✓
10	✓	✓	✓	✓	✓	✓	✓
16	✓	✓	✓	✓	✓	✓	✓
16T	✓	✓		✓	✓	✓	✓
C[1]			✓				
N	✓	✓	✓	✓	✓	✓	✓
8T	✓	✓	✓	✓	✓		
SS	✓		✓		✓		
18	✓	✓	✓	✓	✓	✓	✓

Disponibilités

Cinq médecins ont une indisponibilité par semaines, trois autres ont des disponibilités très réduites, le reste est disponible en tout temps.

Charge de travail

La charge de travail est définie pour tout le personnel par un nombre d'heures minimum et un nombre d'heures maximum à effectuer sur tout l'horizon. De plus, afin d'équilibrer la charge, les médecins ne doivent pas faire plus de 36 heures par semaine.

Règles ergonomiques

- ***Patrons de week-end*** **[ERG1]** : La combinaisons de plusieurs règles de l'hôpital Sacré-Coeur (les règles E06 à E10 du mémoire de Forget [20]), nous permettent de déterminer l'ensemble des patrons autorisés pour les jours du week-end. Il s'agit des règles suivantes :
  - Si un médecin effectue un quart de soir ou de nuit lors d'un vendredi non férié, il doit travailler lors du même quart le lendemain, en considérant l'alternance `cube/choc` s'il y a lieu.

- ◇ Si un médecin effectue un quart quelconque lors d'un vendredi férié, il doit travailler lors du même quart le lendemain, en considérant l'alternance `cube/choc` s'il y a lieu.
- ◇ Si un médecin effectue un quart quelconque lors d'un samedi, il doit travailler lors du même quart le lendemain, en considérant l'alternance `cube/choc` s'il y a lieu.
- ◇ Pour chaque lundi férié, si un médecin complète un quart de jour, c'est parce qu'il a complété un quart de jour le dimanche (en considérant l'alternance `cube/choc`)
- ◇ Si un médecin travaille au cours d'une fin de semaine, il ne doit pas travailler au cours du lundi (non férié) suivant.
- ◇ Si le vendredi précédant un week-end et le lundi suivant sont fériés tous les deux, on considère le lundi comme un jour régulier.

Notons que dans ce contexte, le week-end commence au quart de soir du vendredi et se termine à la fin du quart de nuit du dimanche. Le tableau 6.18(a) référence les patrons autorisés pour un week-end régulier, le tableau 6.18(b), pour un week-end précédé d'un vendredi férié et le tableau 6.18(c), pour un week-end succédé par un lundi férié.

Tableau 6.17 – Patrons de week-end pour la salle d'urgence de l'Hôpital Sacré-Coeur.

Ven	Sam	Dim	Lun
16	16T	16	congé
16T	16	16T	congé
N	N	N	congé
$\star^1$	8	10	congé
$\star^1$	10	8	congé
$\star^1$	18	18	congé
$\star^1$	congé	congé	$\star^1$

(a) Cas général.

Ven férié	Sam	Dim	Lun
16	16T	16	congé
16T	16	16T	congé
N	N	N	congé
10	8	10	congé
8	10	8	congé
18	18	18	congé
congé	congé	congé	$\star^1$

(b) Cas du vendredi férié.

Ven	Sam	Dim	Lun férié
$\star^1$	16T	16	16T
$\star^1$	16	16T	16
$\star^1$	N	N	N
$\star^1$	8	10	8
$\star^1$	10	8	10
$\star^1$	18	18	18
$\star^1$	congé	congé	congé

(c) Cas du lundi férié.

- ***Patrons de séquence*** **[ERG2]** : Pour respecter un temps de repos minimal entre les quarts, les patrons `soir/jour`, `nuit/jour` et `nuit/soir` sont interdits.
- ***Nombre d'affectations consécutives*** *[ERG3]* **:**
  - Les médecins ne doivent pas travailler plus de quatre jours consécutifs.
  - Les séquences de nuit doivent être de taille trois.
  - Deux blocs de nuits doivent être espacés de 14 jours minimums i.e la taille des séquences de non-nuits doit être supérieure à 14.
- ***Week-ends consécutifs*** **[ERG4]** : Les nouveaux médecins travaillent un week-end sur deux, tandis que les anciens travaillent au plus un week-end sur l'horizon.
- ***Séquences consécutives*** **[ERG5]** :
  - Un médecin peut demander à avoir un minimum de deux jours de congés après un patron `travail/nuit`.
  - Un médecin peut demander d'avoir un minimum de trois jours de congé après une séquence de trois nuits consécutives.

[1] L'étoile représente n'importe quel quart non présent dans la colonne.

- Au retour de vacances, un médecin ne doit pas faire de quart de nuit ni de quarts de type `choc` pendant un minimum de deux jours.

- ***Distribution des quarts importuns*** **[ERG6]** :
  - Les quarts de week-end sont limités à cinq. De plus, si un médecin a plus de 14 jours de vacances pendant l'horizon, il doit obligatoirement travailler au cours d'un week-end.
  - Les quarts de `nuit` sont limités à trois par horizon.
  - Les cours sont limités à un par horizon.
  - Les quarts `18` sont limités à quatre par horizon.
  - Les quarts `18` sont limités à un par semaine.
  - Les quarts `SS` sont limités à deux par semaine.
- ***Distribution entre plusieurs types de quarts*** **[ERG7]** : Les rapports entre le nombre de quart de `jour` et le nombre de quarts de `soir` des horaires individuels doivent correspondre au nombre total de quart de `jour` disponibles sur le nombre total de quarts de `soir` disponibles. De même pour les rapports entre les quarts `cubes` et les quarts `chocs`. (plus de quarts de `jour` que de quarts de `soir` ? Plus de quarts `cube` que de quarts `choc`).

Règles d'équité

- ***Équilibrage d'un type de quart*** **[FAI]** :
  - Équilibrage des quarts `18`.
  - Équilibrage des quarts `SS`.
  - Équilibrage des réunions.
  - Équilibrage des quarts de type `nuit`.
  - Équilibrage du nombre d'heures hebdomadaire

6.1.6 Hôpital Santa-Cabrini (HSCa)

Cette description s'inspire du chapitre cinq du mémoire de maîtrise de Forget [20]. S'y référer pour plus de détails.

Description générale

Tableau 6.18 – Description générale de la salle d'urgence de l'Hôpital Santa-Cabrini.

Nom du fichier :	`SantCabrini.data(2)`
Effectif du personnel :	23 médecins dont 8 nouveaux et 15 anciens
Taille de l'horizon de planification :	1 mois (28 jours)
Nombre de tâches différentes :	8

Description des quarts de travail

Tableau 6.19 – Description des tâches de la salle d'urgence de l'Hôpital Santa-Cabrini.

Symbole	Durée	Début	Fin	Période	Type	Obl/Fac
A	8H	8 :00	16 :00	jour	choc	Obligatoire
B	8H	8 :00	16 :00	jour	cube	Obligatoire
C	8H	16 :00	24 :00	soir	choc	Obligatoire
D	8H	16 :00	24 :00	soir	cube	Obligatoire
E	8H	0 :00	8 :00	nuit	choc	Obligatoire
F	8H	10 :00	18 :00	jour	cube	Facultatif
G	7H	18 :00	1 :00	soir	cube	Facultatif
H	8H	6 :00	14 :00	jour	cube	Facultatif

Description de la demande

Tous les quarts doivent être réalisés chaque jour sauf le quart H qui n'a pas lieu le dimanche.

Disponibilités

Seulement deux médecins sont disponibles en tout temps.

Charge de travail

La charge de travail est limitée par un nombre d'heures maximum par semaine variable selon les disponibilités. De plus chaque médecin spécifie un nombre d'heure

qu'il souhaite effectuer au cours de l'horizon. La priorité doit être donnée aux médecins les plus disponibles : Forget [20] définit à cet effet plusieurs groupes de médecins.

Règles ergonomiques

- ***Patrons de week-end*** **[ERG1]** : Dans ce contexte, un médecin qui effectue un quart quelconque le samedi doit effectuer ce même quart le dimanche en respectant l'alternance cube/choc s'il y a lieu. Les patrons autorisés sont donc : `A/B`, `B/A`, `C/D`, `D/C`, `E/E`, `F/F`, `F/off`, `off/F`, `off/off`.
- ***Patrons de séquence*** **[ERG2]** : Pour respecter un temps de repos minimal entre les quarts, les patrons `soir/jour`, `nuit/jour` et `nuit/soir` sont interdits (rotation circadienne).
- ***Nombre d'affectations consécutives*** **[ERG3]** :
  - La plupart de médecins ne doivent pas travailler pendant plus de quatre jours consécutifs.
- ***Distribution des week-ends*** **[ERG4]** : Sauf exception, la fréquence des week-ends travaillés est 1/3.
- ***Séquences consécutives*** **[ERG5]** :
  - Un médecin peut demander d'avoir un minimum de trois jours de congé après une séquence de trois nuits consécutives.
- ***Distribution des quarts importuns*** **[ERG7]** :
  - Sauf exception, le nombre de quarts de week-end est limité à quatre par horizon.
  - Sauf exception, le nombre de quarts de nuit est limité à deux par horizon.
- ▷ ***Préférences*** **[ERG8]**.

6.1.6.1 Règles d'équité

- **[FAI]** : Équilibrer la charge de travail en respectant la priorité des groupes (Forget [20]).

6.1.7 Hôpital Général Juif (HGJ)

Cette description s'inspire du mémoire de maîtrise de Buzòn [7]. S'y référer pour plus de détails.

Description générale

Tableau 6.20 – Description générale de la salle d'urgence de l'Hôpital Général Juif.

Nom du fichier :	`HGJ.data(2)`
Effectif du personnel :	22 médecins
Taille de l'horizon de planification :	13 semaines (91 jours)
Nombre de tâches différentes :	13

Description des quarts de travail

Tableau 6.21 – Description des quarts de semaine de la salle d'urgence de l'Hôpital Général Juif.

Symbole	Durée	Début	Fin	Période
D1	9H	8 :00	17 :00	jour
D2	8H	8 :00	16 :00	jour
S	4H	8 :00	12 :00	jour
FT	8H	8 :00	16 :00	jour
P	8H	12 :00	20 :00	midi
E1	7H	16 :00	23 :00	soir
E2	8H	16 :00	24 :00	soir
NT	9H	23 :00	8 :00	nuit

Tableau 6.22 – Description des quarts de week-end de la salle d'urgence de l'Hôpital Général Juif.

Symbole	Durée	Début	Fin	Période
X1	8H	8 :00	16 :00	jour
X2	8H	8 :00	16 :00	jour
Y1	8H	16 :00	24 :00	soir
Y2	8H	16 :00	24 :00	soir
Z	8H	0 :00	8 :00	nuit

Description de la demande

Les quarts `D1`, `S`, `FT`, `P`, `E1`, `E2` et `NT` doivent être réalisés chaque jour de la semaine Les quarts `D1`, `D1`,`D1`, `D1` et `D1` doivent être réalisés chaque jour du week-end. Le quart `D2` est uniquement présent au cours des Lundis.

Disponibilités

Charge de travail

Dans ce contexte, la charge de travail est calculée en nombre de quarts. Dans le contrat de chaque médecin est spécifié un nombre de quarts moyen par semaine, un nombre de quarts de nuit par horizon et un nombre de quarts de week-end par horizon. Voici les trois règles de type **`[WOR]`** à satisfaire :

- Respecter la charge moyenne de travail par semaine (en nombre de quarts).
- Respecter le nombre de quarts de nuit par horaire.
- Respecter le nombre de quarts de fin de semaine par horaire.

Règles ergonomiques[1]

- ▷ ***Patrons de week-ends*** **`[ERG1]`** :. Un médecin peut choisir de ne pas avoir de week-end brisé.
- ▷ ***Patrons de séquences*** **`[ERG2]`** : Pour chacune des préférences suivantes, nous précisons les patrons autorisés ou interdits :
  - ▷ ***Blocs de nuit homogènes*** : On ne souhaite pas avoir des blocs mélangeant des nuits avec un autre type de quart de travail. On interdit donc les patrons suivant :

    `nuit/jour`,
    `jour/nuit`,
    `nuit/midi`,
    `midi/nuit`,
    `nuit/soir`,
    `soir/nuit`.
  - ▷ ***Blocs de travail homogènes*** : On souhaite que chaque bloc de travail soit homogène à l'un des types `jour`, `midi`, `soir` ou `nuit`. On autorise donc

[1]Dans le mémoire de Buzòn [7], toutes ces règles sont considérées comme souples du fait de l'utilisation d'une méthode tabou.

uniquement les patrons suivant :

`jour/non-travail`,
`non-travail/jour`,
`midi/non-travail`,
`non-travail/midi`,
`soir/non-travail`,
`non-travail/soir`,
`nuit/non-travail`,
`non-travail/nuit`.

- ▷ ***Rotation circadienne au sein des blocs de travail.*** On souhaite que dans un même bloc de travail, l'enchaînement des quart respecte la rotation circadienne `jour/midi/soir/nuit`. On autorise donc tous les patrons qui permettent une transition vers le type `non-travail` (c'est-à-dire tous les patrons de la règle précédente), ainsi que les patrons `jour/midi`, `midi/soir` et `soir/nuit`.
- ▷ ***Rotation circadienne de blocs de travail homogènes.*** On souhaite que les blocs de travail soient homogènes à l'un des types `jour`, `midi`, `soir` ou `nuit` et qu'en outre les blocs s'enchaînent en respectant la rotation circadienne. Pour ce faire, on autorise les patrons binaires permettant les transitions vers le type `non-travail` (règle sur les blocs de travail homogènes) ainsi que les patrons ternaires suivants :

  `jour/congé/nuit`,
  `jour/congé/midi`,
  `midi/congé/midi`,
  `midi/congé/soir`,
  `soir/congé/soir`,
  `soir/congé/nuit`,
  `nuit/congé/nuit`,
  `nuit/congé/jour`.

- ▷ ***Nombre d'affectations consécutives*** **[ERG3]** :
  - ▷ Les séquences de travail sont bornées.
  - ▷ Les séquences de quarts de nuit sont bornées.
  - ▷ Deux blocs de nuits doivent être espacés de quatorze jours minimum i.e la taille des séquences de non nuits doit être supérieur à quatorze.
  - ▷ Les jours de congé isolés sont à éviter : la taille minimale des séquences de

jours de congé est deux.

- ▷ Les quarts de travail isolés sont à éviter : la taille minimale des séquences de travail est deux.

▷ ***Week-ends consécutifs.*** **[ERG4]** : La distribution des week-ends est laissée au choix du médecin qui peut décider de travailler un week-end sur deux.

▷ ***Séquences consécutives*** **[ERG5]** : Garantir un minimum de deux jours de congés après une séquence de quarts de nuit.

6.2 Modélisation avec HIBISCUS

6.2.1 Cas des unités de soins infirmiers

HIBISCUS permet de modéliser toutes les règles à l'exception des suivantes :

- ▷ les préférences et aversions (**[ERG8]**),
- ▷ la minimisation des écarts par rapport à la demande cible de chaque période (**[DEM]**) ainsi que
- ▷ le respecter de la distribution entre les quarts de types `jour`, `soir` et `nuit` (**[ERG7]**).

Pour cette dernière, il n'a pas non plus été possible de la prendre en compte sous la forme d'une contrainte dure (en donnant une tolérance sur le nombre de quarts de chaque type) car la plupart des problèmes deviennent alors incohérents.

6.2.2 Cas des salles d'urgence de médecins

Nous avons simplifié tous ces contextes en ne tenant pas compte des jours fériés.

6.2.2.1 Hôpital Sacré-Coeur

Parmi les règles ergonomiques, certaines n'ont pu être modélisées : **[E13]**[1], **[E15]**[1] et la contrainte **[E16]**[1] qui fait le lien entre les horizons successifs. Les règles d'équité ont quant à elle été modélisées grossièrement par des contraintes dures.

Dans ce contexte, les règles d'équité servent de critère d'évaluation. Cependant, comme les problèmes générés sont incohérents (la demande ne peut être satisfaite à cause du personnel trop peu nombreux), nous avons ajouté un médecin fictif ne

[1] Dans la nomenclature de Forget [20].

pouvant faire qu'un maximum de huit quarts H sur l'horizon. Ces quarts, s'ils sont affectés au médecin fictif, devront être pourvus par une equipe volante.

6.2.2.2 Hôpital Santa-Cabrini

Ce contexte est bien plus simple que le précédant. Toutes les règles ont pu être modélisées. L'objectif visant à satisfaire au mieux les charges souhaitées par le personnel ne peut quant à lui pas être pris en compte directement. Nous avons donc fixé une charge minimum en fonction du souhait de chaque médecin et de ses disponibilités.

6.2.2.3 Hôpital Général Juif

Pour ce contexte, nous n'avons pas encore conçu de jeux de données reflétant les données réelles. Il est assez difficile à modéliser avec **HIBISCUS** car la plupart des règles sont souples.

6.3 Choix des stratégies

6.3.1 Cas des unités de soins infirmiers

Pour ces contextes, nous utilisons des stratégies basées sur une décomposition par horaires individuels (voir section 5.2.2.1). Pour chacune d'elle nous choisissons un ordonnancement des membres du personnel aléatoire. Ensuite nous selectionnons les cinq configurations suivantes qui diffèrent seulement de par les choix de variable et de valeur :

[DpH1] choix de variable lexicographique et choix de valeur par l'heuristique de la section 5.1.2,

[DpH2] choix de variable selon le principe du *first fail* et choix de valeur par l'heuristique de la section 5.1.2,

[DpH3] choix conjoint de variable et de valeur par l'heuristique de la section 5.1.2,

[DpH4] choix de variable lexicographique et choix de valeur aléatoire et enfin,

[DpH5] choix de variable selon le principe du *first fail* et choix de valeur aléatoire.

Pour les stratégies concernées, l'heuristique de la section 5.1.2 est définie uniquement avec les heuristiques d'incitation associées à la satisfaction de la demande (contrainte verticales).

6.3.2 Cas des salles d'urgence de médecins

Pour ces contextes, nous utilisons des stratégies basées sur une décomposition par jours (voir section 5.2.2.2). Nous avons selectionné cinq configurations qui diffèrent de par l'ordonnancement des jours, les choix de variable et de valeur :

[DpJ1] ordonnancement des week-ends d'abord, choix conjoint de variable et de valeur par l'heuristique de la section 5.1.2,

[DpJ2] ordonnancement des week-ends d'abord, choix de variable lexicographique et choix de valeur par l'heuristique de la section 5.1.2,

[DpJ3] ordonnancement des week-ends d'abord, choix de variable selon le principe du *first fail* et choix de valeur par l'heuristique de la section 5.1.2,

[DpJ4] ordonnancement chronologique, choix de variable selon le principe du *first fail* et choix de valeur par l'heuristique de la section 5.1.2,

[DpJ5] ordonnancement des week-ends d'abord, choix de variable selon le principe du *first fail* et choix de valeur aléatoire.

Pour les stratégies concernées, l'heuristique de la section 5.1.2 est définie avec les heuristiques d'incitation associées aux contraintes de charge de travail et aux contraintes de cardinalité.

6.4 Résultats

6.4.1 Protocole experimental

Pour tester la robustesse de nos stratégies, nous allons déterminer tous les choix arbitraires à l'aide de générateurs pseudo-aléatoires. Ces choix arbitraires sont esentiellement l'ordonnancement statique des sous-problèmes pour les stratégies DpHx et les bris d'égalité (méthode pour départager les couples variable/valeur qui se font attribuer un poids identique par les heuristiques d'incitation). Nous allons effectuer dix tests sur chaque fichier de données avec des germes aléatoires différents. Le temps maximum est fixé à une heure. Si aucune solution n'a été trouvée en ce temps, le test est considéré comme un échec. Pour chaque test réussi, nous conservons le temps d'exécution ainsi que le nombre de retours en arrière. Les tableaux présentés dans cette section contiennent les moyennes (m) et les écarts types (σ) des ces grandeurs calculées sur les tests réussis. Les détails de ces tests sont disponibles à l'annexe A.

Détails techniques

Le programme **HIBISCUS** est codé en langage C++ et utilise la bibliothèque de programmation par contraintes *Ilog Solver v4.4*. Les tests ont été réalisés sur des machines Sun Ultra-10 à 440Mhz avec 1Go de mémoire vive et utilisant le système d'exploitation Sun Solaris 2.8.

6.4.2 Unités de soins infirmiers

La première chose que nous allons comparer est la robustesse des stratégies. D'une part, les stratégies donnent-elles des résultats sur l'ensemble des fichiers de données et d'autre part, pour un même fichier, quelles sont leurs taux de réussite (table 6.23). On

Tableau 6.23 – Nombre de succès sur dix tests.

Stratégies	DpH1	DpH2	DpH3	DpH4	DpH5
CHILD	**8**	6	7	0	0
ERMGH	**8**	6	5	7	**8**
BC	**7**	4	0	4	5
DIA	4	1	0	**10**	1

remarque que la stratégie DpH1 est la plus robuste. Le seul cas ou elle n'obtient pas le meilleur résultat est pour le fichier DIA pour lequel la demande est quasi inexistante. Comme les seules heuristiques d'incitation prises en compte pour les stratégies (DpH1, DpH2 et DpH3) sont basées sur la demande, cela explique leur mauvais résultats (les heuristiques ne guident pas). La bonne performance de l'heuristique DpH4 pour ce même fichier s'explique aussi du fait que le problème est pratiquement découplé.

En comparant les résultats de DpH1 avec ceux de DpH4 et DpH5 (et en faisant abstraction du fichier DIA), on constate la supériorité de la stratégie guidée par l'heuristique.

En comparant DpH1, DpH2 et DpH3, on constate la supériorité du choix de variable (qui correspond au choix du jour dans ce cas) lexicographique. De façon intuitive on pouvait s'attendre en effet à ce que remplissage d'un horaire individuel soit plus facile dans l'ordre chronologique, étant donné les nombreuses contraintes de consécutivité.

Tableau 6.24 – Moyennes et écarts-types des temps d'exécution.

Stratégies	**DpH1**		**DpH2**		**DpH3**	
Fichiers	m	σ	m	σ	m	σ
CHILD	**9.45**	14.77	71.43	146.07	650.08	1016.64
ERMGH	78.29	180,24	7.07	3.60	886.00	873.51
BC	**8.44**	4.75	23.55	14.53	-	-
DIA	**14.15**	6.56	-	-	-	-

(a) Temps d'exécution pour les stratégies utilisant les heuristiques.

Stratégies	**DpH4**		**DpH5**	
Fichiers	m	σ	m	σ
CHILD	-	-	-	-
ERMGH	**4.07**	0.65	4.73	2.53
BC	14.46	22.82	75.96	121.01
DIA	20.19	14.01	-	-

(b) Temps d'exécution pour les stratégies utilisant un choix de valeur aléatoire.

Pour ce qui est de temps de calcul (table 6.24) et du nombre de retours en arrière (table 6.25), là encore la stratégie DpH1 donne globalement les meilleurs résultats avec des temps en dessous de cent secondes et une dispersion relativement faible.

Tableau 6.25 – Moyennes et écarts-types des nombre de retours en arrières (en milliers).

Stratégies	**DpH1**		**DpH2**		**DpH3**	
Fichiers	m	σ	m	σ	m	σ
CHILD	**3.59**	7.04	44.60	95.63	231.24	360.00
ERMGH	23.97	53.57	1.57	1.07	3013.01	295.52
BC	**1.35**	1.77	5.73	4.58	-	-
DIA	**6.03**	3.29	-	-	-	-

(a) Nombre de retours arrière pour les stratégies utilisant les heuristiques.

Stratégies	**DpH4**		**DpH5**	
Fichiers	m	σ	m	σ
CHILD	-	-	-	-
ERMGH	**1.18**	0.38	1.53	1.39
BC	0.72	0.25	14.46	22.82
DIA	8.76	6.48	-	-

(b) Nombre de retours arrière pour les stratégies utilisant un choix de valeur aléatoire.

6.4.3 Salles d'urgences de médecins

Tableau 6.26 – Nombre de succès sur 10 tests.

Stratégies	**DpJ1**	**DpJ2**	**DpJ3**	**DpJ4**	**DpJ5**
HSCa	**10**	**10**	**10**	4	9
HSCo	9	**10**	**10**	4	4

Que ce soit en termes de robustesse (table 6.26), de temps de calcul (table 6.27) ou de nombre de retours en arrière (table 6.28), les trois stratégies DpJ1, DPJ2 et DpJ3 se

démarquent par rapport aux deux autres. Les mauvais résultats de la stratégie DpJ4 montrent l'importance de privilégier l'affectation des variables de week-end au début. Enfin les résultats mitigés de la stratégie DpJ5 montrent l'avantage des heuristiques que ce soit pour le choix de valeur uniquement (DpJ2 & DpJ3) ou pour le choix conjoint de variable et de valeur (DpJ1).

Dans le cas des stratégies travaillant jour par jour, il n'y a plus avantage à suivre l'ordre lexicographique pour le choix des variables puisqu'il s'agit en fait d'un choix de personne, que la seul contrainte liant les personnes est la demande et qu'elle ne tient pas compte de l'ordre. Pour départager ces trois stratégies il faudrait d'avantage de fichiers de données car aucune ne se démarque vraiment sur ces tests.

Tableau 6.27 – Moyennes et écarts-types des temps de résolution.

Stratégies	**DpJ1**		**DpJ2**		**DpJ3**	
Fichiers	m	σ	m	σ	m	σ
HSCa	**0.44**	0.05	**0.39**	0.03	**0.64**	0.63
HSCo	26.82	37.75	46.32	80.50	**4.42**	5.24

Stratégies	**DpJ4**		**DpJ5**	
Fichiers	m	σ	m	σ
HSCa	640.50	898.32	15.52	16.96
HSCo	104.65	117.82	451.36	746.60

Sur un exemplaire HSCo similaire, Saadie [41] obtient une solution en un peu plus d'une seconde après 69 retours en arrière avec une stratégie différente de la notre qui consiste à affecter les jours de week-ends en premier, puis les nuits en semaine et enfin les jours et les soirs en semaine. Nos meilleures stratégies font aussi bien en moyenne en ce qui concerne le temps de résolution, mais avec un nombre de retours en arrière bien moins élevé. Pour ce qui est du nombre de quarts ouverts (quarts prévus qui ne sont pas couverts par l'horaire), le solution de Saadie en compte quatre. Nous en obtenons suivant les solutions entre deux et huit. Sans faire office de réelle comparaison entre les programmes, cela permet de donner un point de repère sur les performances.

Tableau 6.28 – Moyennes et écarts-types du nombre de retours en arrières.

Stratégies	DpJ1		DpJ2		DpJ3	
Fichiers	m	σ	m	σ	m	σ
HSCa	**1.00**	1.26	**0.80**	0.98	26.40	68.24
HSCo	711.50	806.04	1058.90	1822.06	**71.60**	127.23

Stratégies	DpJ4		DpJ5	
Fichiers	m	σ	m	σ
HSCa	177844.25	275065.01	1098.67	1222.73
HSCo	7953.75	9360.20	111215.25	191450.10

6.5 Discussion

6.5.1 Modélisation avec HIBISCUS

Nous avons montré dans ce chapitre la capacité de **HIBISCUS** à modéliser des contextes bien différents, même si ce n'est pas toujours avec exactitude. La principale lacune est la modélisation des règles souples qui sont toujours présentes dans un problème de confection d'horaire surtout quand il s'agit d'introduire l'équité.

Il existe des extensions au formalisme de CSP qui prennent en compte directement des contraintes souples : il s'agit des formalismes de PCSP et de HCOP, respectivement "Partial Constraint Satisfaction Problem" et "Hierarchical Constraint Optimisation Problem". Si l'on veut conserver le formalisme de CSP, trois techniques s'offrent à nous :

- l'ajout au modèle de contraintes dures légèrement relaxées,
- l'utilisation d'une fonction objectif ou
- l'utilisation des heuristiques.

La première technique est très peu satisfaisante. En effet, l'ajout de contraintes dures rend souvent le problème incohérent. Souvent il faut rechercher à tâtons le bon degré de relaxation pour chaque contrainte souple. La méthode devient trop interactive.

La seconde technique est plus classique. Elle consiste à mettre dans un objectif des variables contraintes représentant les grandeurs à optimiser conjointement.

Par expérience, l'algorithme de branch-and-bound classique de la programmation par contrainte peine à trouver de bonnes solutions sans de bonnes heuristiques.

La dernière technique semble bien plus satisfaisante et cadre parfaitement avec notre travail sur les heuristiques de recherche modulaires. Pour chaque contrainte souple on peut associer une heuristique d'incitation. Pour l'instant nous avons développé des heuristiques pour :

- les contraintes de cardinalités,
- les contraintes sommes et
- les préférences.

Dans les expérimentations de la section 6.4, nous nous sommes concentrés sur la réalisabilité des solutions lors du guidage par les heuristiques. Nous aurions pu tenir compte de certaines règles souples en ajoutant d'autres heuristiques. Cependant, l'accumulation des heuristiques d'incitation soulève un autre problème : celui de la calibration. Bien connu pour être difficile, on le retrouve dans beaucoup d'autres approches comme la recherche tabou et plus généralement en programmation multi-critères.

Finalement, l'utilisation conjointe des heuristiques, d'une fonction objectif et de procédures de recherche avancées telles que LDS ("Limited Discrepancy Search") [21] et DDS ("Depth-bounded Discrepancy Search") [46] s'inscrit tout à fait dans la continuité du projet actuel.

6.5.2 Critique des résultats

Tout d'abord, nos conclusions sont à tempérer par le fait que notre base de test est relativement réduite. Cependant, les données sont suffisamment hétérogènes pour montrer qu'**HIBISCUS** est un programme flexible et robuste. On peut s'interroger sur la robustesse d'une méthode qui dépendamment d'un paramètre sans signification réel (le germe du générateur pseudo-aléatoire) peut ne donner aucun résultat. Les meilleures stratégies ont tout de même un taux de réussite supérieur à 0.8. De plus, rien n'empêche l'utilisateur de relancer le programme quand la résolution devient trop longue. Cette procédure pourrait même être automatisée en prenant en paramètre un temps de résolution maximum.

On peut en outre déplorer le manque de référence et de comparaison avec d'autres méthodes sur les mêmes données. Cela aurait permis de savoir si les exemplaires de

fichier de la base de test sont des problèmes faciles ou difficiles. En plus de la comparaison avec le programme de Saadie [41], nous avions prévu initialement de comparer nos résultats avec ceux du programme IRIS [25, 28], mais quelques différences entre les modèles nous ont empêché de le faire. Pour indication, nous donnons quand même une comparaison des coûts des solutions obtenues sur l'un des fichiers de test. Il s'agit du fichier CHILD_juillet2001 sur lequel nous obtenons un coût moyen de 552.75 en moins de dix secondes tandis que IRIS trouve une première solution de coût 208 en vingt minutes et une solution de coût 130 en à peu près huit heures. Une différence de coût en faveur de IRIS était attendue, étant donné que **HIBISCUS** ne tient pas compte dans ses heursistiques de la valeur de la fonction objectif de IRIS. Cependant, comme nous le mentionnons plus haut, aucune des solutions générées par IRIS n'est réalisable pour **HIBISCUS**. Ces premiers chiffres sont donc à relativiser.

CONCLUSION

Dans ce mémoire, nous avons présenté une méthode de confection d'horaires en milieu hospitalier utilisant la programmation par contraintes. Les contributions de ce travail sont multiples et interviennent à plusieurs niveaux.

Tout d'abord, l'étude des contextes de plusieurs hôpitaux de la région de Montréal nous a permis d'identifier un noyau de règles générales que l'on retrouve dans des environnements aussi différents que des unités de soins infirmiers et des salles d'urgence de médecins. Nous avons décrit un modèle de programmtion par contraintes concis de l'ensemble des règles en discutant pour chacune de la performance des algorithmes de filtrage engendrés. Le modèle intègre en outre un grand nombre de contraintes globales dont les récentes `STRETCH` et `PATTERN`.

Nous avons également décrit un cadre général permettant de concevoir des stratégies de recherche adaptées à tout type de problème concret que l'on peut extraire du modèle. Nous avons discuté des différentes façons de décomposer le problème et avons proposé une structure modulaire originale pour nos heuristiques. Cette dernière contribution dépasse le cadre de la confection d'horaire et constitue un apport non négligeable pour la programmation par contraintes qui souffre d'un manque certain de littérature en ce qui concerne les heuristiques de recherche. Cette structure repose sur le concept d'*heuristique d'incitation* qui représente la brique de base de toute heuristique. Nous avons fourni des heuristiques d'incitation pour deux types de contraintes très largement représentées dans les problèmes de confection d'horaires : les contraintes de cardinalité et les contraintes sommes.

Enfin, nous avons décrit et modélisé avec **HIBISCUS** plusieurs contextes réels et hétérogènes que nous avons utilisés comme base de test pour comparer différentes stratégies de résolution. Les résultats obtenus sont encourageant : **HIBISCUS** donne des solutions en moins de cent secondes.

HIBISCUS se révèle être très flexible, s'adaptant bien à plusieurs situations. De plus il permet en outre de définir simplement des stratégies adaptées à des catégories spécifiques de problèmes. Cependant, **HIBISCUS** n'est pas encore utilisable en pratique car il ne permet pas de modéliser de façon systématique les contraintes souples, indispensables pour exprimer convenablement l'équité, par exemple. Nous proposons déjà quelques éléments de réponse et il en ressort que la structure des heuristique de **HIBISCUS** pourrait être utilisée pour pallier ce problème. Voici une liste (non exhaustive) des développement futurs :

- Intégrer au modèle des fonctions d'évaluations et transformer **HIBISCUS** en une méthode d'optimisation.
- Concevoir plus d'heuristiques d'incitation notamment pour les règles souples.
- Tester le mécanisme des heuristiques sur des problèmes moins complexes afin d'en affiner le fonctionnement.
- Comparer **HIBISCUS** avec d'autres approches (génération de colonnes [25, 28], recherche tabou [7] et programmation linéaire en nombres entiers [4, 20]).

BIBLIOGRAPHIE

[1] S. Abdennadher et H. Schenker. Nurse scheduling using constraint logic programming. In *Eleventh Annual Conference on Innovative Applications of Artificial Intelligence, IAAI-99, Orlando, Florida.* AAAI Press, 1999.

[2] Acme Express Inc. *docs2000.* http ://www.docs2000.net/.

[3] AtStaff, Inc. *Physician Scheduler.* http ://www.msisoftware.com/.

[4] H. Beaulieu. Planification de l'horaire des médecins dans une salle d'urgence. Mémoire de maîtrise, Université de Montréal, 1998.

[5] N. Beldiceanu et E. Contejean. Introducing global constraints in CHIP. *Mathematical and Computer Modelling*, 12 :97–123, 1994.

[6] R. N. Burns et M. W. Carter. Work force size and single shift schedules with variable demands. *Management Science*, 31(5) :599–607, 1985.

[7] I. Buzòn. La confection des horaires de travail des médecins d'urgence résolue à l'aide de la recherche tabou. Mémoire de maîtrise, Université de Montréal, 2001.

[8] G. Cangini. A constraint programming local search algorithm for physician scheduling. Rapport Technique C7PQMR PO2000-26-X, Centre de Recherche sur les Transports, Canada, Juin 2000.

[9] A. Caprara, M. Monaci, et P. Toth. Models and algorithms for a staff scheduling problem (to appear). *Mathematical Programming*, 2003.

[10] M. W. Carter et S. D. Lapierre. Scheduling emergency room physicians. *Health Care Management Science*, 4 :347–360, 2001.

[11] Y. Caseau, P. Giullo, et E. Levenez. A deductive and object-oriented approach to a complex scheduling problem. In *Third International Conference, DOOD'93, Phoenix, Arizona, USA, December 6-8, 1993, Proceedings*, pages 67–80. Springer, 1993.

[12] Celayix Inc. *eTime Xpress.* http ://www.celayix.com/etimeoverview.html.

[13] B. M. W. Cheng, J. H. M. Lee, et J. C. K. Wu. A constraint-based nurse rostering system using a redundant modeling approach. In *Eighth IEEE International Conference on Tools with Artificial Intelligence*, pages 140–148. IEEE Computer Society Press, 1996.

[14] COSYTEC. *Gymnaste.* http ://www.cosytec.com/constraint-programming/casesstudies/health-care.htm.

[15] G. B. Dantzig. A comment on eddie's traffic delays at toll booths. *Operations Research*, 2 :339–341, 1954.

[16] S. J. Darmoni, A. Fajner, N. Mahé, A. Leforestier, M. Vondracek, O. Stelian, et M. Baldenweck. Horoplan : computer-assisted nurse scheduling using contraint-based programming. *Journal of the Society for Health Systems*, 5 :41–54, 1995.

[17] K. A. Dowsland. Nurse scheduling with tabu search and strategic oscillation. *European Journal of Operation Research*, 106(2-3) :393–407, 1998.

[18] F. Fages et L. Akash. A global constraint for cutset problems. In *CP-AI-OR 2003, Fifth International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 153–165, 2003.

[19] W. Feuilletin. Génération automatique d'horaires d'infirmières à l'aide de la programmation par contraintes. Rapport Technique C7PQMR PO2000-32-X, Centre de Recherche sur les Transports, Canada, Octobre 2000.

[20] F. Forget. Confection automatisé des horaires de médecins dans une salle d'urgence. Mémoire de maîtrise, Université de Montréal, 2002.

[21] W.D Harvey et M. L. Ginsberg. Limited discrepancy search. In Chris S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95); Vol. 1*, pages 607–615, Montréal, Québec, Canada, August 20-25 1995. Morgan Kaufmann, 1995.

[22] K. Heus. *Gestion des plannings infirmiers, Application des techniques de programmation par contraintes.* Thèse de doctorat, Université Joseph Fourier - Grenoble 1, Mai 1996.

[23] Interactive Information R&D. *Effective Computer Aided Scheduling.* http ://www.i2rd.com/ECAS/.

[24] InTime. *SpeedShift.* http ://www.intimesoft.com/.

[25] B. Jaumard, F. Semet, et T. Vovor. A generalized linear programming model for nurse scheduling. *European Journal of Operational Research*, 107 :1–18, 1998.

[26] P. Knauth. The design of shift systems. *Ergonomics*, 36 :15–27, 1993.

[27] P. Knauth. Design better shift systems. *Ergonomics*, 27 :1 :39–44, 1996.

[28] P. Labit. IRIS : Amélioration d'une méthode de génération de colonnes pour la confection d'horaires d'infirmières. Mémoire de maîtrise, École Polytechnique de Montréal, 2000.

[29] Y. Lebbah et O. Lhomme. Accelerating filthering techniques for numeric CSPs. *Artificial Intelligence*, 139 :109–132, 2002.

[30] Madrigal Soft Tools. *ResSched.* http ://www.madrigalsoft.com/.

[31] A. Meisels, E. Gudes, et G. Solotorevsky. Combining rules and constraints for employee timetabling. *Internaltional Journal of Intelligent Systems*, 12 :419–439, 1997.

[32] H. Meyer auf' m Hofe. Solving rostering tasks by generic methods for constraint optimization. *International Journal of Foundations of Computer Science*, 12(5) :671–693, 2001.

[33] H. E. Miller, W. P. Pierskalla, et G. J. Rath. Nurse scheduling using mathematical programming. *Operations Research*, 24 :857–870, 1976.

[34] U. Montanari. Networks of constraints : Fundamental properties and application to picture processing. *Information Sciences*, 7 :95–132, 1974.

[35] Open Wave. *ShiftTrack.* http ://www.open-wave.com/.

[36] G. Pesant. A filtering algorithm for the stretch constraint. In T. Walsh, editor, *Principles and Practice of Constraint Programming - CP 2001*, pages 183–195. Springer, 2001.

[37] G. Pesant. The pattern constraint (in prepartation). 2003.

[38] L. M. Rousseau, G. Pesant, et M. Gendreau. A hybrid algorithm to solve a physician rostering problem. In *Second Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Paderborn, Germany*, 2000.

[39] J.-C. Régin. A filtering algorithm for constraints of difference in CSPs. In *Proc. of AAAI-94*, pages 362–367, Seattle, WA, 1994.

[40] J.-C. Régin. Generalized arc consistency for global cardinality constraints. In *Proc. of AAAI-96*, pages 209–215. AAAI Press/MIT Press, 1996.

[41] M. Saadie. Planification de l'horaire des médecins dans une salle d'urgence par la programmation par contraintes. Mémoire de maîtrise, Université de Montréal, 2003.

[42] ScheduleSource, Inc. *ScheduleSource TeamWork.* http ://www.schedulesource.com/.

[43] G. Thompson. A simulated-annealing heuristic for shift scheduling using non-continuously available employees. *Computer Operational Research*, 23(3) :275–288, 1996.

[44] P. Van Hentenryck, V. Saraswat, et Y. Deville. Design, implementation, and evaluation of the constraint language cc(FD). *Journal of logic programming*, 37 :139–164, 1998.

[45] P. Vilím et R. Barták. Filtering algorithms for batch processing with sequence dependent setup times. In *Proceedings of the 6th International Conference on AI Planning and Scheduling, AIPS'02*, 2002.

[46] T. Walsh. Depth-bounded discrepancy search. In *Proceedings of the sixteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 1388–1395. Morgan Kaufmann, 1997, 1997.

[47] D. M. Warner. Scheduling nursing personnel according to nursing preference : A mathematical programming approach. *Operations Research*, 24 :842–856, 1976.

[48] Yardley Software. *I.M.P.S.* http ://yardleysoftware.com/.

ANNEXE A

Détail des résultats

A.1 Fichier CHILD_juillet2001

Tableau A.1 – Fichier CHILD_juillet2001 - Stratégie NBN FU MEM (DpH1)

Germe	succés	#backtracks	Temps (s)	Valeur[1]
0	✓	1127	4.3	573
1	✓	761	3.9	557
2	✓	848	3.6	491
3	✓	22221	48.5	522
4	✓	1144	4.4	588
5	✓	521	2.8	499
6	✓	1022	3.5	558
7	✗	-	> 3600	-
8	✓	1110	4.6	634
9	✗	-	> 3600	-
Moyenne[2]		3594.25	9.45	552.75
Écart-type[2]		7043.16	14.77	44.65

Tableau A.2 – Fichier CHILD_juillet2001 - Stratégie NBN MiDS MEM (DpH2)

Germe	succés	#backtracks	Temps (s)	Valeur[1]
0	✓	1648	5.1	505
1	✗	-	> 3600	-
2	✓	370	2.6	490
3	✗	-	> 3600	-
4	✓	1019	4.4	546
5	✗	-	> 3600	-
6	✗	-	> 3600	-
7	✓	3161	8.9	562
8	✓	258417	398	588
9	✓	2959	9.6	546
Moyenne[2]		44595.67	71.43	539.50
Écart-type[2]		95628.92	146.07	33.12

Tableau A.3 – Fichier CHILD_juillet2001 - Stratégie NBN MEM (DpH3)

Germe	succés	#backtracks	Temps (s)	Valeur[1]
0	✓	23821	71.7	656
1	✓	688839	1824	835
2	✓	2072	7.8	714
3	✓	1268	6.7	665
4	✓	898052	2619.3	665
5	✗	-	> 3600	-
6	✓	2079	8.7	750
7	✗	-	> 3600	-
8	✗	-	> 3600	-
9	✓	2564	12.4	734
Moyenne[2]		231242.14	650.08	718.29
Écart-type[2]		360013.95	1016.64	58.03

A.2 Fichier ERMGH_juin2002

Tableau A.4 – Fichier ERMGH_juin2002 - Stratégie NBN FU MEM (DpH1)

Germe	succés	#backtracks	Temps (s)	Valeur[1]
0	✗	-	> 3600	-
1	✓	1602	5.2	58255
2	✓	7331	20.6	58695
3	✓	1123	4.8	58005
4	✗	-	> 3600	-
5	✓	2058	5.4	58044
6	✓	165403	554.7	56897
7	✓	1455	4.9	58982
8	✓	1213	4.4	60395
9	✓	11587	26.3	60911
Moyenne[2]		23971.50	78.29	58773.00
Écart-type[2]		53573.25	180.24	1232.91

Tableau A.5 – Fichier ERMGH_juin2002 - Stratégie NBN MiDS MEM (DpH2)

Germe	succés	#backtracks	Temps (s)	Valeur[1]
0	✗	-	> 3600	-
1	✗	-	> 3600	-
2	✓	627	3.3	61251
3	✓	658	4.2	60835
4	✗	-	> 3600	-
5	✓	1808	9.3	59451
6	✗	-	> 3600	-
7	✓	3586	9.6	59755
8	✓	648	3.4	59232
9	✓	2076	12.6	62152
Moyenne[2]		1567.17	7.07	60446.00
Écart-type[2]		1076.10	3.60	1052.86

Tableau A.6 – Fichier ERMGH_juin2002 - Stratégie NBN MEM (DpH3)

Germe	succés	#backtracks	Temps (s)	Valeur[1]
0	✗	-	> 3600	-
1	✗	-	> 3600	-
2	✓	680075	2192	52320
3	✓	187581	582	50484
4	✗	-	> 3600	-
5	✗	-	> 3600	-
6	✗	-	> 3600	-
7	✓	10067	32	53087
8	✓	626741	1611	52463
9	✓	2076	13	54444
Moyenne[2]		301308.00	886.00	52560.60
Écart-type[2]		295521.92	873.51	1280.93

Tableau A.7 – Fichier ERMGH_juin2002 - Stratégie NBN FU R (DpH4)

Germe	succés	#backtracks	Temps (s)	Valeur[1]
0	✗	-	> 3600	-
1	✓	1075	4.1	44794
2	✗	-	> 3600	-
3	✓	978	3.7	46615
4	✓	1258	4.4	45114
5	✓	815	3.2	45096
6	✓	786	3.3	49196
7	✓	1336	4.8	48948
8	✗	-	> 3600	-
9	✓	2004	5	42498
Moyenne[2]		1178.86	4.07	46037.29
Écart-type[2]		387.24	0.65	2223.07

Tableau A.8 – Fichier ERMGH_juin2002 - Stratégie NBN MiDS R (DpH5)

Germe	succés	#backtracks	Temps (s)	Valeur[1]
0	✓	5102	11.2	43630
1	✓	738	3.2	45046
2	✓	1644	5	47235
3	✓	701	3.2	44374
4	✗	-	> 3600	-
5	✗	-	> 3600	-
6	✓	660	3.2	47158
7	✓	898	3.4	46178
8	✓	1265	4.1	43868
9	✓	1257	4.5	43392
Moyenne[2]		1533.13	4.73	45110.13
Écart-type[2]		1386.88	2.53	1461.20

A.3 Fichier BC_sept2001

Tableau A.9 – Fichier BC_sept2001 - Stratégie NBN FU MEM (DpH1)

Germe	succés	#backtracks	Temps (s)	Valeur[1]
0	✓	470	6.0	22264
1	✓	882	7.7	25714
2	✓	1272	8.5	22915
3	✓	537	6.0	25214
4	✓	403	5.6	23014
5	✗	-	> 3600	-
6	✓	5626	19.8	26316
7	✗	-	> 3600	-
8	✗	-	> 3600	-
9	✓	310	5.5	22414
Moyenne[2]		1350.71	8.44	23978.71
Écart-type[2]		1770.07	4.75	1578.91

Tableau A.10 – Fichier BC_sept2001 - Stratégie NBN MiDS MEM (DpH2)

Germe	succés	#backtracks	Temps (s)	Valeur[1]
0	✗	-	> 3600	-
1	✗	-	> 3600	-
2	✓	959	8.2	20462
3	✗	-	> 3600	-
4	✓	13031	46.5	24964
5	✗	-	> 3600	-
6	✓	5969	24.9	22065
7	✓	2946	14.6	22264
8	✗	-	> 3600	-
9	✗	-	> 3600	-
Moyenne[2]		5726.25	23.55	22439.75
Écart-type[2]		4579.16	14.53	1616.68

Tableau A.11 – Fichier BC_sept2001 - Stratégie NBN FU R (DpH4)

Germe	succés	#backtracks	Temps (s)	Valeur[1]
0	✗	-	> 3600	-
1	✗	-	> 3600	-
2	✓	1055	7.7	20314
3	✓	586	5.7	22215
4	✓	399	5.1	20915
5	✗	-	> 3600	-
6	✓	837	6.6	22364
7	✗	-	> 3600	-
8	✗	-	> 3600	-
9	✗	-	> 3600	-
Moyenne[2]		719.25	14.46	21452.00
Écart-type[2]		248.45	22.82	865.64

Tableau A.12 – Fichier BC_sept2001 - Stratégie NBN MiDS R (DpH5)

Germe	**succés**	**#backtracks**	**Temps (s)**	**Valeur**[1]
0	✗	-	> 3600	-
1	✗	-	> 3600	-
2	✓	2207	12.0	21016
3	✓	1068	7.9	27064
4	✓	2783	17.1	22364
5	✗	-	> 3600	-
6	✓	6282	25.1	18314
7	✓	59975	317.7	23264
8	✗	-	> 3600	-
9	✗	-	> 3600	-
Moyenne[2]		14.46	75.96	22404.40
Écart-type[2]		22822.48	121.01	2867.45

A.4 Fichier DIA_sept2001

Tableau A.13 – Fichier DIA_sept2001 - Stratégie NBN FU MEM (DpH1)

Germe	succés	#backtracks	Temps (s)	Valeur[1]
0	✓	3061	8.0	29829
1	✓	9103	22.8	31429
2	✗	-	> 3600	-
3	✗	-	> 3600	-
4	✗	-	> 3600	-
5	✓	9517	18.2	30930
6	✗	-	> 3600	-
7	✗	-	> 3600	-
8	✓	2445	7.6	30630
9	✗	-	> 3600	-
Moyenne[2]		6031.50	14.15	30704.50
Écart-type[2]		3288.98	6.56	580.47

Tableau A.14 – Fichier DIA_sept2001 - Stratégie NBN MiDS MEM (DpH2)

Germe	succés	#backtracks	Temps (s)	Valeur[1]
0	✗	-	> 3600	-
1	✗	-	> 3600	-
2	✗	-	> 3600	-
3	✗	-	> 3600	-
4	✗	-	> 3600	-
5	✗	-	> 3600	-
6	✗	-	> 3600	-
7	✓	1604657	2812	25878
8	✗	-	> 3600	-
9	✗	-	> 3600	-
Moyenne[2]		1604657	2812	25878
Écart-type[2]		0	0	0

Tableau A.15 – Fichier DIA_sept2001 - Stratégie NBN FU R(DpH4)

Germe	succés	#backtracks	Temps (s)	Valeur[1]
0	✓	3287	8.5	27378
1	✓	3813	10.2	26627
2	✓	2251	6.3	29278
3	✓	17310	32.2	28879
4	✓	5429	12.68	24778
5	✓	4109	9.9	26628
6	✓	2712	6.3	24978
7	✓	13515	39.2	26580
8	✓	18483	35.5	25028
9	✓	16716	41.1	23928
Moyenne[2]		8762.50	20,19	26408.20
Écart-type[2]		6478.01	14.01	1678.28

A.5 Fichier HSCa

Tableau A.16 – Fichier HSCa - Stratégie DBD wef MEM (DpJ1)

Germe	succés	#backtracks	Temps (s)
0	✓	1	0.4
1	✓	0	0.5
2	✓	0	0.4
3	✓	4	0.5
4	✓	0	0.4
5	✓	0	0.4
6	✓	1	0.4
7	✓	2	0.5
8	✓	0	0.5
9	✓	2	0.4
Moyenne[2]		1.00	0.44
Écart-type[2]		1.26	0.05

Tableau A.17 – Fichier HSCa - Stratégie DBD wef FU MEM (DpJ2)

Germe	succés	#backtracks	Temps (s)
0	✓	1	0.4
1	✓	1	0.4
2	✓	0	0.4
3	✓	0	0.4
4	✓	0	0.3
5	✓	0	0.4
6	✓	2	0.4
7	✓	0	0.4
8	✓	3	0.4
9	✓	1	0.4
Moyenne[2]		0.80	0.39
Écart-type[2]		0.98	0.03

Tableau A.18 – Fichier HSCa - Stratégie DBD wef MiDS MEM (DpJ3)

Germe	succés	#backtracks	Temps (s)
0	✓	0	0.4
1	✓	230	2.5
2	✓	1	0.4
3	✓	1	0.4
4	✓	1	0.4
5	✓	1	0.4
6	✓	25	0.7
7	✓	1	0.4
8	✓	1	0.4
9	✓	3	0.4
Moyenne[2]		26.40	0.64
Écart-type[2]		68.24	0.63

Tableau A.19 – Fichier HSCa - Stratégie DBD lex MiDS MEM (DpJ4)

Germe	succés	#backtracks	Temps (s)
0	✓	3	0.5
1	✓	58920	389
2	✓	8	0.5
3	✗	-	>3600
4	✗	-	>3600
5	✗	-	>3600
6	✓	652446	2172
7	✗	-	>3600
8	✗	-	>3600
9	✗	-	>3600
Moyenne[2]		177844.23	640.50
Écart-type[2]		275065.01	898.32

Tableau A.20 – Fichier HSCa - Stratégie DBD wef MiDS R (DpJ5)

Germe	succés	#backtracks	Temps (s)
0	✓	44	1.1
1	✓	1612	22.6
2	✗	-	>3600
3	✓	1990	27.6
4	✓	2930	41.4
5	✓	3020	41.9
6	✓	36	0.6
7	✓	20	0.5
8	✓	227	3.5
9	✓	9	0.5
Moyenne[2]		1098.67	15.52
Écart-type[2]		1222.73	16.96

A.6 Fichier HSCo

Tableau A.21 – Fichier HSCo - Stratégie DBD wef MEM (DpJ1)

Germe	succés	#backtracks	Temps (s)	#quarts ouverts[1]
0	✓	197	6.8	8
1	✓	2620	132	8
2	✓	1559	45.7	8
3	✗	-	>3600	-
4	✓	107	4.9	8
5	✓	109	5	8
6	✓	45	2.8	8
7	✓	1145	33.5	8
8	✓	647	17	8
9	✓	686	20.5	8
Moyenne[2]		711.50	26.82	8.00
Écart-type[2]		806.04	37.75	0.00

Tableau A.22 – Fichier HSCo - Stratégie DBD wef FU MEM (DpJ2)

Germe	succés	#backtracks	Temps (s)	#quarts ouverts[1]
0	✓	3013	141	8
1	✓	74	5.6	8
2	✓	62	5.7	8
3	✓	72	5.6	8
4	✓	73	5.6	8
5	✓	53	4.5	8
6	✓	83	6.2	8
7	✓	1029	23.7	8
8	✓	285	9.3	8
9	✓	5845	256	8
Moyenne[2]		1058.90	46.32	8.00
Écart-type[2]		1822.06	80.50	0.00

Tableau A.23 – Fichier HSCo - Stratégie DBD wef MiDS MEM (DpJ3)

Germe	succés	#backtracks	Temps (s)	#quarts ouverts[1]
0	✓	12	1.8	3
1	✓	7	1.8	3
2	✓	8	1.8	5
3	✓	8	1.8	4
4	✓	330	15.1	3
5	✓	6	1.8	3
6	✓	5	1.8	3
7	✓	322	14.7	3
8	✓	9	1.8	4
9	✓	9	1.8	4
Moyenne[2]		71.60	4.42	3.50
Écart-type[2]		127.23	5.24	0.67

Tableau A.24 – Fichier HSCo - Stratégie DBD lex MiDS MEM (DpJ4)

Germe	succés	#backtracks	Temps (s)	#quarts ouverts[1]
0	✓	23839	303	4
1	✓	58	2.4	4
2	✗	-	>3600	-
3	✗	-	>3600	-
4	✓	5347	80.2	2
5	✗	-	>3600	-
6	✗	-	>3600	-
7	✗	-	>3600	-
8	✗	-	>3600	-
9	✓	2571	686	4
Moyenne[2]		104.65	7953.75	3.50
Écart-type[2]		117.82	9360.20	0.87

Tableau A.25 – Fichier HSCo - Stratégie DBD wef MiDS R (DpJ5)

Germe	succés	#backtracks	Temps (s)	#quarts ouverts[1]
0	✓	16	2.2	7
1	✗	-	>3600	-
2	✓	442815	1744	8
3	✗	-	>3600	-
4	✗	-	>3600	-
5	✗	-	>3600	-
6	✓	424	3.9	8
7	✗	-	>3600	-
8	✗	-	>3600	-
9	✓	1606	55.4	8
Moyenne[2]		451.36	111215.25	7.75
Écart-type[2]		746.60	191450.10	0.43